HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Michael Wolf

Die Analyse des Metatranskriptoms einer Biogasanlage und deren Darstellung auf einer Stoffwechselkarte

2013

Fakultät **Mathematik** / **Naturwissenschaften** / **Informatik**

BACHELORARBEIT

Die Analyse des Metatranskriptoms einer Biogasanlage und deren Darstellung auf einer Stoffwechselkarte

Autor:
Michael Wolf

Studiengang:
Biotechnologie / Bioinformatik

Seminargruppe:
BI10w2-B

Erstprüfer:
Prof. Dr. rer. nat. habil. Röbbe Wünschiers

Zweitprüfer:
Dipl. -Ing. (FH) Lucy Stark

Mittweida, 09 2013

Bibliografische Angaben

Wolf, Michael: Die Analyse des Metatranskriptoms einer Biogasanlage und deren Darstellung auf einer Stoffwechselkarte, 51 Seiten, 29 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Mathematik / Naturwissenschaften / Informatik

Bachelorarbeit, 2013

Dieses Werk ist urheberrechtlich geschützt.

Satz: LaTeX

Referat

Die Bachelorarbeit befasst sich mit dem Zuordnen von Reads eines Datensatzes, der aus aus dem sequenzierten Metatranskriptom von 6 verschiedenen Organismen besteht, zu Genomen ausgewählter Organismen und die Darstellung dieser auf einer Stoffwechselkarte. Dafür wurde ein Programm in Python geschrieben, dass die Gensequenzen der einzelnen Genome mit den Reads des Datensatzes, mit der Hilfe von BLAST, vergleicht. Die Anzahl der Übereinstimmungen spiegelt die Stärke der Expression der einzelnen Enzyme wieder und wurde auf der Stoffwechselkarte „Biochemical Pathways“ graphisch dargestellt. Je nach Trefferanzahl wird das Enzym mit einer bestimmten Farbe hervorgehoben, so dass mit einem flüchtigen Blick Expressionsbereiche und die Stärke dieser leicht zu erkennen sind. Des Weiteren wird die genaue Anzahl der Treffer und die EC-Nummer angezeigt. Neben des Bildes speichert das Programm die Ergebnisse in Textdateien, die tabellarisch aufgebaut sind und deshalb mit einem Tabellenkalkulationsprogramm importiert werden können. Damit nicht jeder Organismus einzeln analysiert werden muss, ist eine Methode geschrieben worden, die mehrere Organismen nacheinander mit dem Datensatz vergleicht, und danach die Ergebnisse miteinander vergleicht. Außerdem wurde eine PROSITE Pattern Suche eingebaut, um EC-Nummern, die in keinem Genom vermerkt sind, in dem Datensatz dennoch zu finden.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

ATP	Adenosinphosphat
CO_2	Kohlenstoffdioxid
CoA	Coenzym A
DNA	Deoxyribonucleic acid
GI	Geneinfo Identifier
GIF	Graphics Interchange Format
HDD	Hard Disk Drive
mRNA	Messenger RNA
Mulit-BLAST	BLAST Durchlauf des Programmes mit mehreren Organismen
nt	Nucleotide
PNG	Portable Network Graphics
RAM	Random Access Memory
RNA	Ribonucleic acid
SQL	Structured Query Language
SVG	Scalable Vector Graphics
W3C	World Wide Web Consortium
XML	Extensible Markup Language

V. Danksagung

Meinen Dank an Prof. Robbe Wünschiers, dass er die Bachelorarbeit ermöglichte, mich betreute und unterstützte, Dipl. Ing.(FH) Lucy Stark für die nette Zusammenarbeit und Betreuung im Labor, sowie für nützliche biotechnologische Hinweise und Tina Giersch B.Sc. für das Bereitstellen von Daten im Praktikum und die nette Zusammenarbeit im Labor. Einen besonderen Dank gilt Gabriel Kind M.Sc. für die Unterstützung beim Erstellen des Programmes, Filtern des Datensatzes, Hilfe beim Schreiben dieser Arbeit mit Latex und sonstige nützliche Tipps.
Außerdem möchte ich der Hochschule, den Professoren und Mitarbeitern danken, dass sie mir dieses interessante Studium ermöglicht haben.

1 Einleitung und Zielstellung

In der modernen Zeit des Internets werden wir Menschen mit Informationen aus aller Welt über alles Mögliche überflutet. Es wird immer schwieriger einen Überblick über alle Daten zu behalten und dabei die Wichtigen von den Unwichtigen zu trennen. Des Weiteren sollten Diese in einen sinnvollen Zusammenhang gebracht, und neue Erkenntnisse daraus abgeleitet werden. Es besteht die Gefahr, dass wichtiges Wissen in Vergessenheit gerät und unwichtiges oder falsches Wissen bestehen bleibt.

Abbildung 1.1: Die Holzbohrassel *Limnoria punctata*.[IMG-1]

Ein Beispiel für ein schlummerndes, jedoch nützliches Wissen, ist die Untersuchung der Holzbohrassel *Limnoria* (siehe Abbildung 1.1). Der ehemalige schrecken der Schifffahrt war berüchtigt für das Durchbohren und Zerstören von Holz. Kaum jemand hat der maritimen Assel Beachtung geschenkt und sich gefragt, wie das Tier es schafft das Holz zu verdauen. Vermutlich hätten die Seefahrer von damals auf die Frage, ob sie die Erforschung ihrer Plage für Sinnvoll erachten, lediglich geantwortet, wie bekämpft man den Schädling. Boyle und Mitchell untersuchten das Insekt und fanden 1978 bei ihren Forschungen heraus, dass sich keine Mikroorganismen in ihrem Verdauungstrakt befinden [Boyle and Mitchell, 1978]. Bei anderen Holz-Essenden Insekten, wie Termiten, helfen Bakterien in den Därmen das Lignin zu verdauen und verwerten. Diese Erkenntnis war zwar ungewöhnlich, wurde jedoch nicht weiter beachtet. Erst 30 Jahre später hat King durch diese Information auf das Vorhandensein von Genen geschlossen, die Enzyme für den Holzabbau kodieren. Bei seinen Versuchen hat er dann Glykosyl-Hydrolysiernde Enzyme nachgewiesen [King et al, 2010]. Durch diese Entdeckung öffneten sich plötzlich neue Möglichkeiten für eine effizientere Umsetzung von ligninhaltigen Pflanzen in beispielsweise Biokraftstoffe. Eine weitere mögliche Anwendung ist der Vorverdau von Biomasse für Biogasanlagen, oder eine weitere Zersetzung

des Gärrestes um mehr Zucker aus den harten Pflanzenteilen zu gewinnen, der dann in Biogas umwandelt wird.

Durch die Digitalisierung und das Einführen von Internationalen Datenbanken wird das Wissen gesammelt und theoretisch für die Ewigkeit gespeichert. Mit gut optimierten Suchalgorithmen lassen sich die Erkenntnisse schnell und einfach abrufen und somit sollten wichtige Informationen nicht mehr in Vergessenheit geraten.

Jedoch bleibt das Problem die Masse an Daten zu Verknüpfen und logische Schlüsse daraus zu ziehen. Speziell die Bioinformatik arbeitet mit einer ungeheuren Masse an Daten.

Das menschliche Genom hat beispielsweise eine Länge von ca. 3,3 Milliarden Basenpaaren, das etwa 21'000 bekannte Gene und ungefähr 48'000 vorhergesagte Gene kodiert [EnsEMBL, 2013]. Wenn das Genom auf eine Computerdatei umgerechnet wird, wäre diese 0,825 GB groß[1], das etwa 165 Musikstücken[2] entspricht. Einfachere Organismen, wie Bakterien, haben nur einen Bruchteil der Erbinformationen eines Menschen. Der Modellorganismus *Escherichia coli K-12* hat ca. 4,6 Millionen Basenpaare, die etwa 4300 Gene kodieren [Blattner et al, 1997].

Um über diese gewaltige Menge an Informationen einen Überblick zu behalten bedarf es der Umwandlung der Rohdaten. Zum Beispiel können die Daten durch das Erstellen von genormten Tabellen sowohl einfacher miteinander verglichen, als auch schneller weiter Verarbeitet werden.

```
ATOM     10  N   VAL A  26      24.210  38.936  -6.067  1.00166.38           N
ANISOU   10  N   VAL A  26    28436  14598  20181  -1891   6000   2058       N
ATOM     11  CA  VAL A  26      23.138  37.951  -5.971  1.00138.27           C
ANISOU   11  CA  VAL A  26    24861  11331  16344  -1611   5541   2029       C
ATOM     12  C   VAL A  26      23.596  36.709  -5.209  1.00125.55           C
ANISOU   12  C   VAL A  26    22582  10080  15042  -1618   5498   1869       C
ATOM     13  O   VAL A  26      24.757  36.308  -5.300  1.00116.91           O
ANISOU   13  O   VAL A  26    21220   8973  14226  -1805   5932   1821       O
ATOM     14  CB  VAL A  26      22.635  37.541  -7.370  1.00131.04           C
ANISOU   14  CB  VAL A  26    24638  10268  14885  -1479   5646   2176       C
ATOM     15  CG1 VAL A  26      21.790  36.280  -7.292  1.00132.47           C
```

Abbildung 1.2: Ausschnitt aus der PDB-Datei 3PQS

Eine sehr wichtige und erfolgreiche Methode, um Daten zu Veranschaulichen, ist das Visualisieren dieser. So kann aus einer PDB Datei, die die geometrischen Daten eines

[1] Erklärung: Ein Nukleotid kann 4 Zustände annehmen, das 2 Bits entspricht. Ein Byte entspricht 8 bits. Ein Gigabyte entspricht 1'000'000'000 byte. => 3,3 nt $\triangleq$ 3,3*2/8 GB = 0,825 GB

[2] Unter der Annahme dass ein Musikstück eine Größe von 5 MB entspricht.

Proteins enthält, ein dreidimensionales Bild erstellt werden, das für den Menschen deutlich anschaulicher ist (siehe Abbildung 1.2 und 1.3). Durch genauere Betrachtung des Bildes können sich neue Erkenntnisse ergeben. So kann durch einfaches Drehen des Proteins ein Kanal sichtbar werden, der dann auf ein Transportprotein hinweist (siehe Abbildung 1.4).

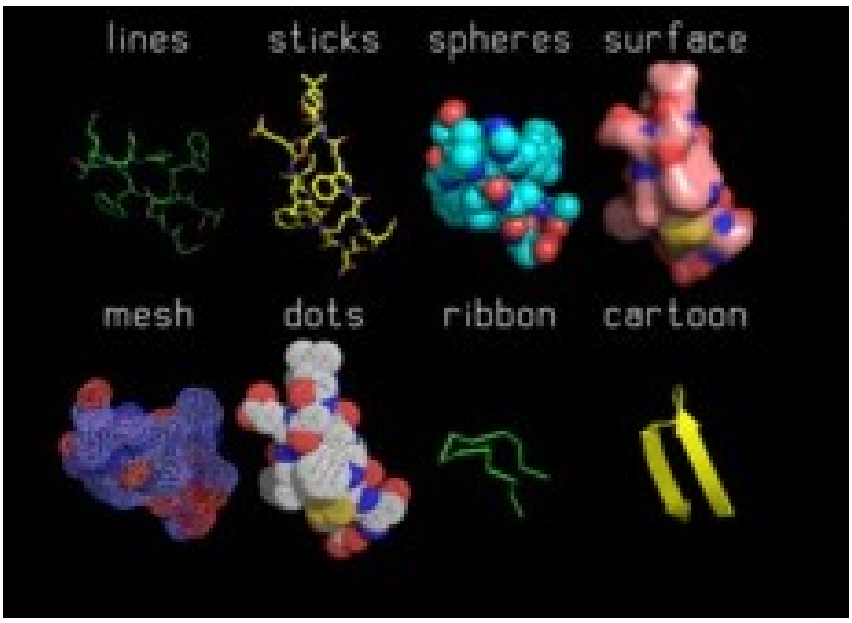

Abbildung 1.3: Verschiedene dreidimensinale Darstellungsformen des gleichen Proteins mit dem Programm PyMOL. [IMG-2]

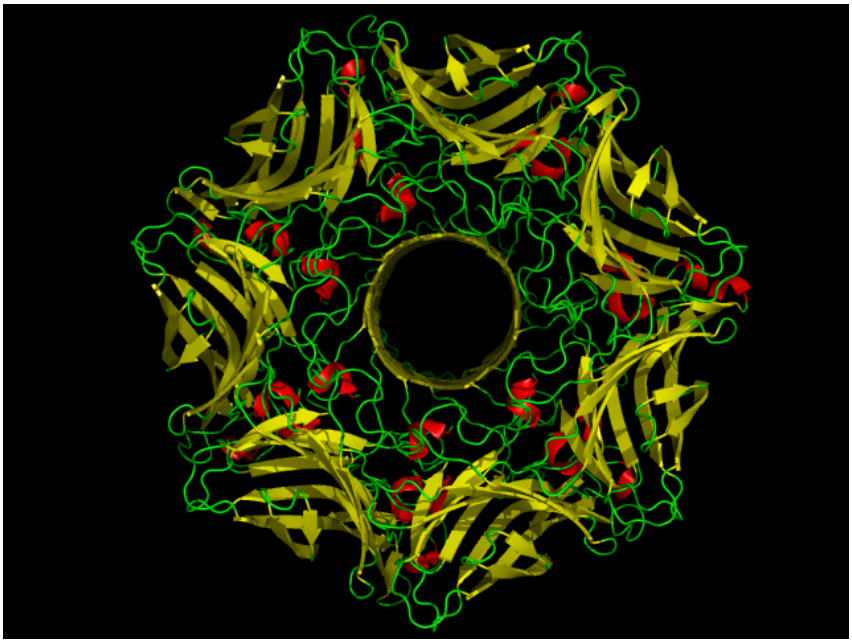

Abbildung 1.4: Das Protein Hemolysin in PyMOL aus der Vogelperspektive. [IMG-3]

Des Weiteren besteht die Möglichkeit, Karten zu zeichnen, um die Zusammenhänge der einzelnen Elemente sichtbar zu machen. In der Biochemie werden gerne Stoffwechselkarten verwendet, um die Verbindungen der Stoffwechselwege darzustellen und dadurch Kreisläufe und Kaskaden besser zu erkennen (siehe Kapitel 2.1 und 2.2). Wenn beispielsweise durch eine Krankheit, ein Element in einem Stoffwechselweg fehlt oder in reduzierter Zahl vorliegt, ist es möglich die Auswirkungen auf andere Stoffwechselwege durch einen schnellen Blick auf die Karte festzustellen.

Die Bachelorarbeit:

> Die Analyse des Metatranskriptoms einer Biogasanlage und deren Darstellung auf einer Stoffwechselkarte

befasst sich mit den oben genannten Problemen. Dabei wird ein großer Datensatz mit einem in Python geschriebenen Programm weiter verarbeitet. Der Datensatz besteht aus Sequenzen, die aus der Sequenzierung der mRNA von 6 verschiedenen Mikroorganismen gewonnen wurde. Das Programm soll die Genome der Organismen gegen den Datensatz BLASTen und das Ergebnis weiter Verarbeiten. Dazu gehört das Darstellen des Vorkommens der Gene der verschiedenen Organismen auf der Stoffwechselkarte. Außerdem sollen Tabellen erstellt werden, die das Analysieren und die weitere Verarbeitung der Ergebnisse erleichtert. Beispielsweise wird eine tabellarische Übersicht über die Verteilung der Treffer auf die jeweiligen Enzymklassen ausgegeben.

2 Grundlagen

Damit die Arbeit besser verstanden wird, werden nun einige Begriffe und Elemente näher erläutert.

2.1 Stoffwechsel

Die Biochemie definiert den Stoffwechsel, auch Metabolismus genannt, als die Gesamtheit von lebenserhaltenden chemischen Transformationen in Zellen von lebenden Organismen. Die einzelnen Stoffwechselwege lassen sich in 2 Abschnitte gliedern, die jedoch fließend ineinander übergehen und vom Vorrat an ATP abhängt. Bei dem Katabolismus (Abbau) werden Nährstoffe in kleinere Bruchstücke zerlegt und zur Energiegewinnung (Gewinndung von ATP) in der Regel vollständig oxidiert. Als Anabolismus (Synthesestoffwechsel) wird die Synthese der niedermolekularen Verbindungen, unter dem Verbrauch von ATP, in die Bausteine[3] und polymeren Makromoleküle[4] der Zelle bezeichnet. Die einzelnen Stoffwechselwege können einfache schnelle Reaktionen, lange Reaktionsketten, oder auch Stoffwechselzyklen sein. Bei der Umwandlung von einem Metaboliten (Zwischenprodukte des Stoffwechsels) zum Nächsten werden spezifische Enzyme benötigt. Enzyme sind biologische Katalysatoren, die die Aktivierungsenergie erniedrigen und dadurch Stoffumwandlungen unter erniedrigten Temperaturen ermöglichen, und die Reaktionszeit deutlich um bis zu 10 Größenordnungen verkürzen[5] [Fuchs, 2007].

Um die Komplexität dieses Systems zu veranschaulichen wird der Citratzyklus näher erläutert (siehe Abbildung 2.1. Der auch Zitronensäurezyklus genannte Stoffwechselweg dient der Oxidation von Acetyl-CoA zu CO_2 unter Abspaltung des Wasserstoffs. Er ist für die Zelle wichtig um Proteine, Fette und Kohlenhydrate abzubauen. Acetyl-CoA wird unter Anderem aus der Oxidierung von Pyrovat, einem Produkt der Glycolyse, gewonnen [Fuchs, 2007]. Der dargestellte Zyklus gilt für die meisten Organsimen, jedoch gibt es verschiedene Bakterien, die einige Abwandlungen davon besitzen.

[3] Bausteine der Zelle: AS, nt, Zuckerphosphate, etc.
[4] Polymere Makromolekple der Zelle: Proteine, Polysaccharide, Polynukleotide, etc.
[5] Faktor 10^{10}: Verkürzung der Halbwertszeit einer Reaktion von 300 Jahren auf eine Sekunde.

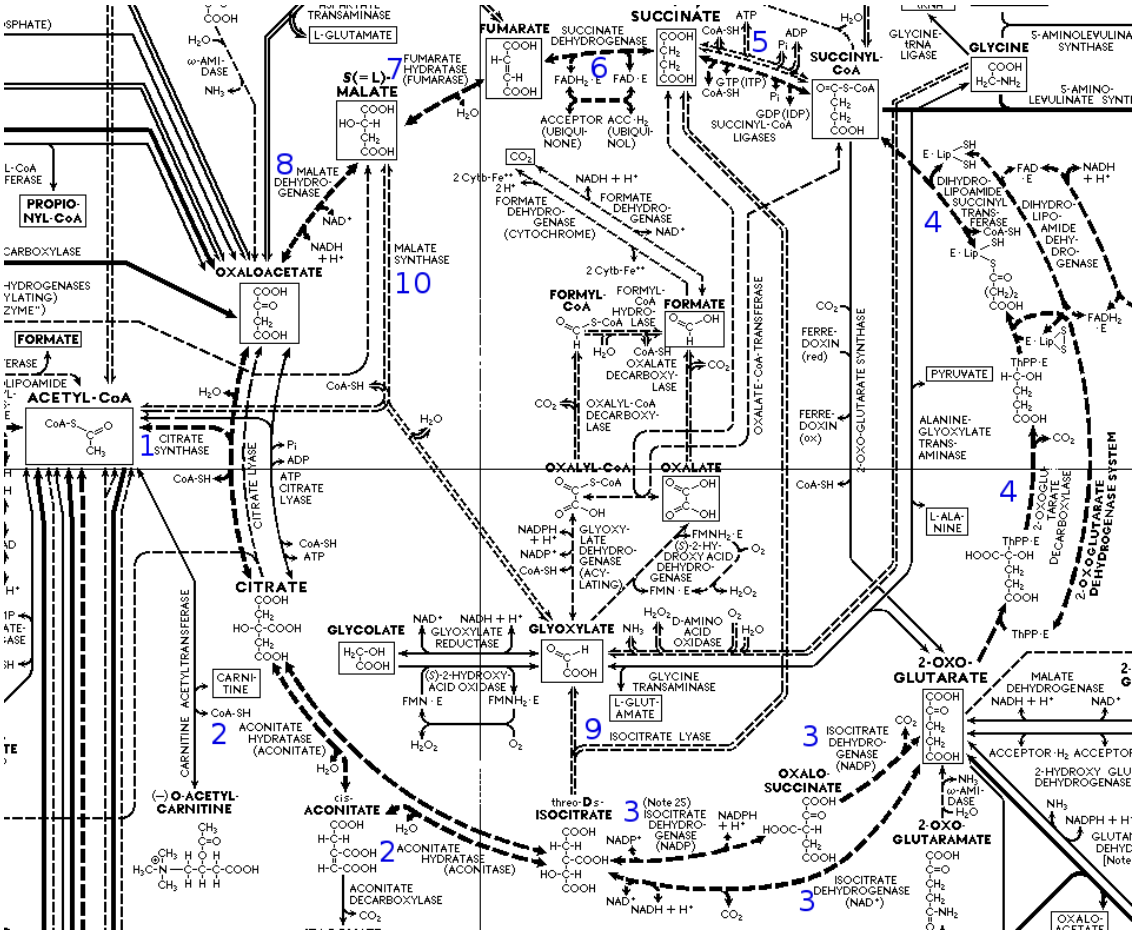

Abbildung 2.1: Der Citratzyklus: 1. Citratbildung, 2. Isomerisierung, 3. und 4. Oxidative Decarboxylierung, 5. Substartkettenphosphorylierung, 6.-8. Regeneration des Oxalacetats durch Dehydrierung, 9. und 10. Glyoxylatzyklus, Bis auf 1.4. sind alle Schritte reversibel.

Bei der Citratsynthase wird Acetyl-CoA zunächst mit Oxalacetat zu Citrat kondensiert, wobei CoA (HS-CoA) irreversibel abgespalten wird. Aconitat-Hydratase katalysiert die reversible Umwandlung der drei Tricarbonsäuren (Citrat -> cis-Aconitat -> Isocitrat) ineinander. Isocitrat-Dehydrogenase katalysiert die Reaktionen, die von Isocitrat zu 2-Oxoglutarat führen. Das zwischenzeitlich gebildete Oxalsuccinat wird sofort decarboxyliert. 2-Oxoglutarat-Dehydrogenase katalysiert eine der Pyruvatdehydrogenase-Reaktion völlig analoge irreversible oxidative Decarbosylierung einer α-Ketosäure. Das entstandene Succinyl-CoA wird durch Succinat-Thiokinase in einer gekoppelten Reaktion, unter Phosphorylierung von Guanosinediphosphat, gespalten. Succinatdehydrogenase oxidiert Succinat zu Fumerat und überträgt die Elektronen auf Ubichinon. Fumerase lagert in einer stereospezifischen Hydratation, die zu L-Malat führt, Wasser an Fumerat an. Malatdehydrogenase dehydrogeniert dann Malat zu Oxalacetat. Die Synthese von Citrat und die Oxidation von 2-Oxoglutarat sind irreversibel, alle anderen Reaktionen sind reversibel. [Fuchs, 2007].

Das Zusammenspiel von 8 verschiedenen Enzymen wird benötigt um Acetyl-CoA abzubauen und etwas Energie (1 Molekül ATP) zu gewinnen. Und dies ist nur ein Stoffwechselweg von Tausenden, die in einer Zelle ablaufen und miteinander vernetzt sind. Fällt ein Enzym aus, kann dies nicht nur den einen Stoffwechselweg, sondern viele Weitere beeinflussen. Damit keine Über- oder Unterproduktion von Metaboliten stattfindet werden die Enzyme von verschiedenen Faktoren, wie beispielsweise Inhibitoren, reguliert. Auf dieses komplexe Thema wird jedoch nicht weiter eingegangen.

2.2 Stoffwechselkarte „Biochemical Pathways“

Die Stoffwechselkarte „Biochemical Pathways “ wurde 1968 von dem Wissenschaftler Dr. Gerhard Michal von Boehringer Mannheim erstellt. Sie wurde auf Basis einer Stoffwechseldatenbank entwickelt, die die meisten bekannten Stoffwechselwege, Metabolismen und Enzyme enthielt. Hauptsächlich wurde sie auf dem menschlichen Metabolom aufgebaut, jedoch befinden sich dort auch Stoffwechselwege anderer Organismen. Die Karte wurde inzwischen mehrfach überarbeitet und erweitert und 1998 als Buchform („Biochemical Pathways: An Atlas of Biochemistry and Molecular Biology“) veröffentlicht. Etwa zur gleichen Zeit wurde sie digitalisiert und über die EXPASY[6] Webseite verfügbar gemacht [Wishart, 2006].

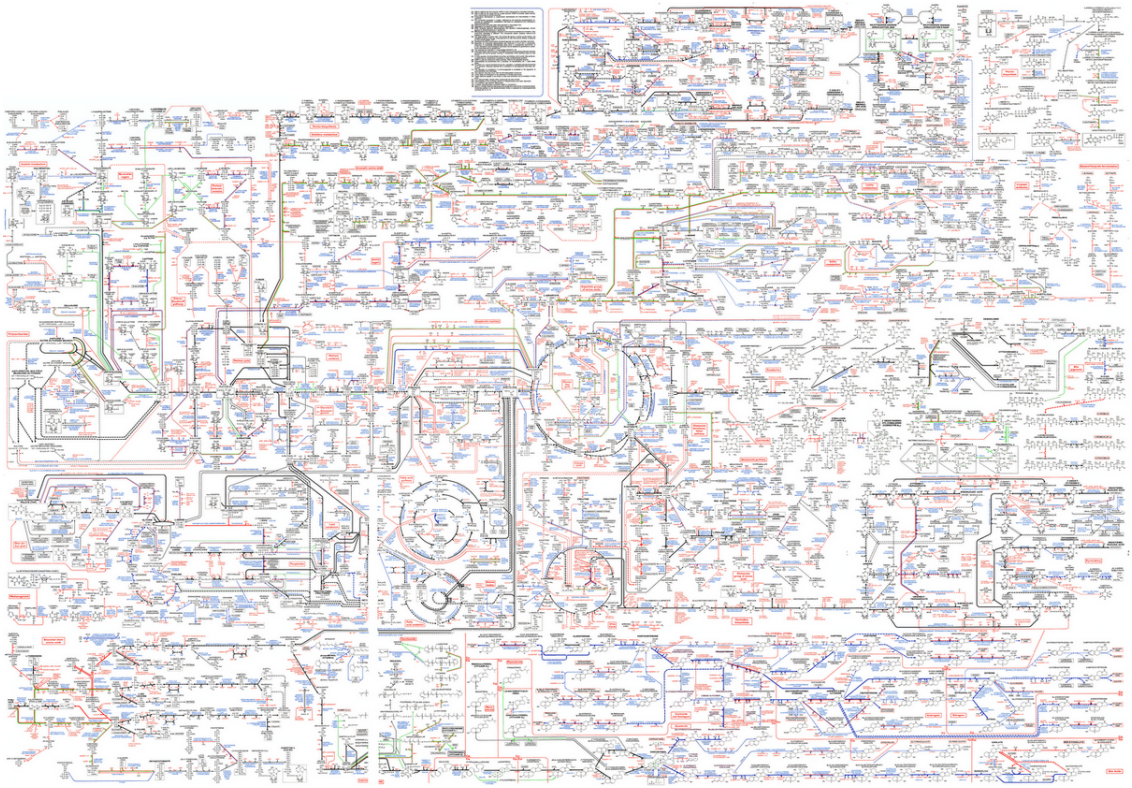

Abbildung 2.2: Die Stoffwechselkarte „Biochemical Pathways “

[6] `http://web.expasy.org/pathways/`

Die Karte ist wie folgt aufgebaut:

- Die Enzymnamen werden blau und mit dem üblichen Namen von der „Enzym Nomenklatur 1984“ und ihre Ergänzungen dargestellt. Enzyme, die dort nicht gelistet sind, werden mit Klammern angezeigt.
- Coenzyme werden rot und andere Reaktionspartner schwarz dargestellt.
- Die Farbe der Pfeile zeigt die Art des Organismus, in dem die Reaktion beobachtet worden ist, oder wahrscheinlich vorkommt:

  – Schwarz - Allgemeine Biochemische Stoffwechselwege
  – Grün - Höhere Pflanzen
  – Blau - Tiere
  – Rot - Einzellige Organismen

  Gestrichelte Pfeile werden für katabolische Stoffwechselwege benutzt, durchgestrichene Linien für Ana- und Amphibolische. Punkte an beiden Enden der Pfeile zeigen Reversibilität und ein orangefarbener Pfeil neben den Linien symbolisiert die bevorzugte Richtung der Reaktion.
- Regulatorische Effekte werden in Orange gezeichnet. Metallionen und ähnliche Aktivatoren befinden sich neben dem Reaktionspfeil. Effektoren mit einer schnellen Durchflussregulation haben einen kontinuierlichen, von einer Seite kommenden orangefarbenen Pfeil. Die langsamen Regulatoren haben jedoch einen gestrichelten orangefarbenen Pfeil. Ein Plus, ebenfalls in Orange, deutet auf eine Erhöhung der Enzymaktivität hin, ein Minus auf eine Erniedrigung. Wenn nur eines von mehreren Enzymen auf diese Art geregelt wird, werden römische Ziffern benutzt. Regulatoren in Klammern sind effektiv nur in einer Gruppe oder Spezies von Organismen vorhanden.
- Coenzyme und Verbindungen mit komplexen Strukturen sind von einem schwarzen Rechteck umrandet, sehr einfache Strukturen jedoch nicht.
- Bei den Endprodukten eines Metabolismus werden die Verbindungsnamen in orangefarbenen Rechtecken gesetzt.
- Für ionisierte Verbindungen wird der Name des Salzes verwendet, um den Enzymnamen zu entsprechen. Die Strukturformeln zeigen jedoch die freie Säure oder Base an, da bei physiologischem pH-Wert häufig verschiedene Stufen der Ionisation gleichzeitig auftreten und deshalb wird die Beteiligung von H^+ oder OH^- nicht gezeigt (mit Ausnahme in Verbindung mit NADH oder NADPH).
- Organischer Phosphor wird mit P abgekürzt, Anorganischer mit Pi und Pyhrophosphat mit PPi.

Auf Grund der Übersichtlichkeit der späteren Darstellung wurde für das Darstellen der Expressionsstärke auf der Boehringer Map eine Karte in Graustufen benutzt.

2.3 BLAST

Wenn zwei oder mehrere Protein- oder Nukleotidsequenzen miteinander verglichen werden sollen, gibt es in der Bioinformatik verschiedene Programme, die dieser Aufgabe mit ihren Vor- und Nachteilen nachkommen. Basic Local Alignment Search (BLAST) ist ein heuristischer Algorithmus, der verwendet wird um lokale Sequenzübereinstimmungen schnell und effizient zu finden. Dieser Algorithmus ist in verschiedenen Tools (BLASTn, BLASTp, psi-BLAST, etc.) integriert, die auf ihre Einsatzgebiete hin optimiert wurden. Beispielsweise wird bei BLASTn, das auf Nukleotidsequenzen spezialisiert ist, das Programm DUST[7] verwendet, um Regionen mit geringer Komplexität herauszufiltern und bei BLASTp (auf das Vergleichen von Proteinen spezialisiert) das Programm SEG[8]. Auf der BLAST Webseite[9] gibt es eine Übersicht über die Programme mit Erklärungen und Hilfestellungen, sowie eine Webintegration zur Benutzung dieser. [Madden, 2002]

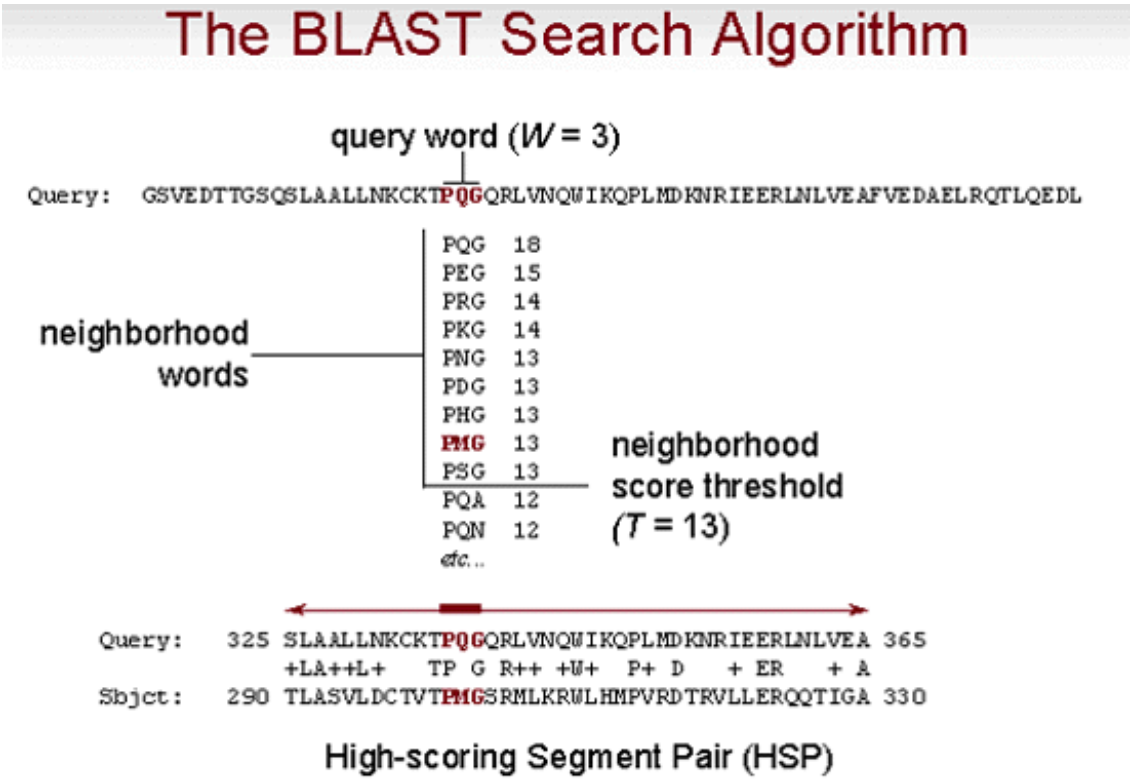

Abbildung 2.3: Der BLAST Algorithmus vereinfacht dargestellt.[IMG-4]

[7] ftp://ftp.ncbi.nlm.nih.gov/pub/agarwala/windowmasker/windowmasker_suppl.pdf
[8] https://www.ncbi.nlm.nih.gov/books/NBK62051/def-item/BLAST_glossary.SEG
[9] http://BLAST.ncbi.nlm.nih.gov/BLAST.cgi

Der BLAST Algorithmus lässt sich wie folgt Gliedern: [Mount, 2004]

1. Entfernen von Regionen mit geringer Komplexität:
   Regionen mit geringer Komplexität, die wenig verschiedene Elemente enthalten (zB. viele A Nukleotide hintereinander), können einen hohen Score ergeben obwohl sie eine geringe Signifikanz haben. Dadurch verwirren sie das Programm und sollten maskiert werden. Wie oben beschrieben werden dafür verschiedene Programme, je nach Sequenzart, verwendet.
2. Erstellen einer Liste mit k-Buchstaben:
   Von der Eingabesequenz werden alle möglichen Wörter mit k hintereinander folgenden Buchstaben erstellt und in eine Liste geschrieben. Bei einer DNA Sequenz ist k normalerweise 11. Z.B. mit der Eingabesequnez ABCDEF und k = 3 werden die Wörter: ABC, BCD, CDE und DEF gebildet.
3. Erweitern der Liste mit Hilfe einer Matrix:
   In diesem Schritt wird die Liste aus Punkt 2 mit ähnlichen Wörtern erweitert. Dabei wird ein Wort aus der Liste genommen und mit allen möglichen Wörtern der Wortlänge verglichen (z.B. ABC mit AAA, AAB, AAC, etc.) und der Score mit Hilfe einer Matrix berechnet. BLAST verwendet häufig die Blossum62 Matrix, kann aber auch andere wie PAM120 verwenden. Wenn der Score über dem “Nachbarschafts-Schwellwert“ liegt, wird das ähnliche Wort in die Liste aufgenommen. Dadurch wird die Signifikanz der BLAST Suche erhöht, da nicht nur nach identischen, sondern auch nach ähnlichen Sequenzen gesucht wird.
4. Ordnen der Liste in einen effizienten Suchbaum:
   Damit das Programm schneller arbeitet, wird aus der Liste ein effizienter Suchbaum erstellt.
5. Durchsuchen der Datenbank nach exakten Übereinstimmungen:
   Die Wörter aus der Liste, bzw. aus dem Suchbaum werden mit den Sequenzen der Datenbank verglichen und nach identischen Übereinstimmungen gesucht.
6. Erweitern der Treffer zu „high-scoring segment pair“ (HSP):
   Bei einer exakten Übereinstimmung wird das Wort in beide Richtungen erweitert und der Score neu berechnet. Der Erweiterungsvorgang findet so lange statt, bis der Score unter dem Grenzwert liegt und das neue (längere) Wort wird in die Liste der HSP eingetragen.
7. Verbinden von zwei oder mehr HSP Regionen:
   In der Liste der HSPs gibt es Treffer, die sich überlappen, oder kaum auseinander liegen. Diese werden miteinander verbunden, wenn der nach dem Verknüpfen berechnete Score über dem Schwellwert liegt. Diese Berechnung wird mit der Poisson Methode vorgenommen, die das Paar mit dem größten kleineren Wert bevorzugt (Bei zwei kombinierten HSP Regionen mit den Paarwerten (79, 56) und (70, 60) wird (70, 60) genommen, da 60>56).
8. Evaluieren der Treffer:
   Am Schluss werden die gefunden Treffer auf ihre Signifikanz geprüft. Dabei wird der E-Wert jeder Übereinstimmung berechnet und gegen einen Schwellwert ver-

glichen. Ist der berechnete Wert darüber, wird das HSP ausgegeben, wenn nicht, dann wird der Vergleich verworfen.
Der E-Wert gibt die Anzahl der Treffer wieder, die erwartet werden, wenn die Datenbank nach dem Treffer durchsucht wird. Der Wert verringert sich exponentiell, wie sich der Score der Übereinstimmung steigert. Folglich gibt ein geringer E-Wert eine hohe Signifikanz des Treffers wieder. Durch das Festlegen des E-Wertes als Schwellwert können die zufälligen und weniger signifikanten Treffer herausgefiltert werden.

In dieser Arbeit wird das neue Programmpacket BLAST2 verwendet, und aus diesem die Programme „BLASTn“ für das Vergleichen der Nukleotidsequenzen, sowie „makeBLAST“ zum Erstellen einer Datenbank aus einer Fasta-Datei.

2.4 PROSITE

PROSITE[10] ist eine Datenbank von Proteinfamilien und Domänen. Gegenwärtig (Stand: 22.07.2013) hat PROSITE 1670 dokumentierte Einträge, die eine eindeutige PROSITE ID besitzen (zB. PS50020), 1308 Pattern und 1065 Profile. Sie basiert auf der Beobachtung, dass die große Anzahl der Proteine auf Basis ihrer Gemeinsamkeiten zu einer begrenzten Menge von Familien gruppiert werden kann. Proteine oder Proteindomänen, die zu einer bestimmten Familie gehören, teilen sich normalerweise funktionelle Eigenschaften und kommen von einem gemeinsamen Vorfahren. [Sigrist et al, 2013]

Beim Studium der Proteinfamilien wird offensichtlich, dass machen Regionen während der Evolution besser konserviert sind als Andere. Diese Regionen sind üblicherweise wichtig für die Funktion oder Aufrechterhaltung der 3D Struktur des Proteins. Durch das Analysieren von konstanten und variablen Bereichen bei einer Gruppe von ähnlichen Sequenzen ist es möglich eine Signatur für eine Proteinfamilie oder Domäne zu erlangen, die sie von den anderen Proteinen abgrenzt. Mit Hilfe dieser Signaturen ist es Möglich neue und unbekannte Proteine zu einer spezifischen Familie zuzuordnen und Vorhersagen über seine Funktion zu treffen [Sigrist et al, 2002].

Die Signatur wird üblicherweise in Form eines Patterns wiedergegeben. Dieses ist wie folgt aufgebaut:

Beispiel: <[AC](3)-x-V-x(2,4)-ED.

- Es wird der Standard IUPAC Ein-Buchstaben Code für Aminosäuren verwendet.
- X entspricht einer beliebigen Aminosäuren.
- Verschiedene Aminosäuren an der gleichen Position sind in einer eckigen Klammer [].

[10] `http://PROSITE.expasy.org/`

- Eine geschweifte Klammer wird verwendet, wenn alle Aminosäuren an dieser Position vorkommen dürfen, außer diejenigen in der Klammer.
- Die Elemente sind mit einem '-' voneinander getrennt.
- Wiederholungen eines Elementes kann man durch die Zahl der Wiederholungen in einer runden Klammer () hinter dem Element setzen.
- Wenn die Anzahl der Wiederholungen nicht genau sind, wird der Bereich in runden Klammern angegeben, wobei Start und Ende durch ein Komma getrennt sind.
- Wenn das Pattern auf das N oder C Terminale Ende begrenzt ist, sind vor bzw. hinter dem Pattern < oder > zu setzen.
- Ein Punkt beendet das Pattern.

Für das Durchsuchen des Datensatzes mit PROSITE Pattern wurde das Pearl Script ps_scan.pl verwendet, dass von der Webseite `ftp://ftp.expasy.org/databases/PROSITE/ps_scan/` heruntergeladen werden kann.

3 Material und Methoden

3.1 Verwendete Dateien, Tools und Programme

In diesem Abschnitt möchte der Autor kurz erläutern welche Programme, Dateien und Tools er verwendet hat, und warum er sich für diese entschied.

3.1.1 Python

Python[11] ist eine moderne mächtige interpretierte High-Level Programmiersprache, die 1991 von Guildo van Rossum eingeführt worden ist. Sie enthält Objekte, Module, Ausnahmeregelungen, Threads und ein automatisches Speichermanagement. Des Weiteren ist Python Plattformunabhängig, das durch das Erzeugen des Bytecodes ermöglicht wird. Außerdem können C oder C++ Blöcke in Python integriert werden, sowie Python Code in C oder C++. Der nächste Vorteil ist, dass Python Open Source ist, was zur Folge hat, dass die Sprache stetig und schnell erweitert wird sowie eine große Community besitzt. Die Sprache ist schnell und einfach zu lernen, da sie unter anderem eine High-Level Programmiersprache ist, und sich der Programmierer nicht um die Low-Level Details (wie Speichermanagement) kümmern muss. Weiter ist die Sprache dynamisch aufgebaut, das dem Anwender das Festlegen der Variablen abnimmt. Die Standardbibliothek enthält viele nützliche Bibliotheken wie die GUI Bibliothek Tkinter, das ein schnelles entwerfen einer Graphischen Oberfläche ermöglicht. [Jackson, 2012]

Es gibt sicher noch weitere Vorteile, bei denen das persönliche Befinden eine Rolle spielt, doch ein paar Nachteile sollten auch genannt werden. Durch die dynamische Programmierung und dem automatischem Speichermanagement ist Python eine Arbeitsspeicherhungrige Sprache, das bei älteren Rechnern mit wenig RAM zu Problemen führen kann. Außerdem ist sie langsamer im Vergleich zu kompilierten Sprachen wie C oder C++, da der Python Code nur interpretiert wird und dabei lediglich in Bytecode und nicht in Maschienensprache umgewandelt wird. [Jackson, 2012]

Das Programm „Mapper“ der Bachelorarbeit soll eventuell in das Cyanofactory Projekt[12] integriert werden. Das Warehouse CyanofactoryKB basiert auf Python und um das Portieren zu vereinfachen wurde die gleiche Sprache für das Programm verwendet. Ein weiter Grund ist die Plattformunabhängigkeit. Das Programm muss auf OS X (Apple) für den Gebrauch auf dem internen Server laufen, sowie unter Kubuntu (Linux), da auf diesem Betriebssystem die Entwicklung von Statten ging und Windows (Microsoft), da-

[11] http://www.python.org/

[12] Masterarbeit *CyanoFactory KB: Umsetzung einer Knowledge Base für das Forschungsprojekt CyanoFactory* von Gabriel Kind

mit weitere Personen das Programm benutzen können, ohne das Betriebssystem zu wechseln. Außerdem war der Umstieg von Java auf Python einfach, da einiges Wissen von Java auch in Python verwendet werden konnte, und Python einige Verbesserungen mit sich brachte, die das Programmieren angenehmer machten.

3.1.2 Externe Module

Mit der Standardbibliothek von Python sind viele Dinge möglich, jedoch fehlen einige Funktionen für bestimmte Anwendungsfälle. Um sich unnötige Programmierarbeit zu sparen, wurden weitere externe Module und Bibliotheken integriert[13]:

BioPython
Das BioPython Projekt formierte sich 1999 als eine Gemeinschaft, die bioinformatische Open Source Programme, die in Python geschrieben sind, sammelte oder selbst erstellte. Es basiert auf dem erfolgreichen BioPearl Projekt, will die Bibliotheken jedoch für Personen verfügbar machen, die mit Python Programmieren. Heute ist BioPython verbreitet und unter Wissenschaftlern anerkannt. Die Bibliotheken werden stetig erweitert und neue Funktionen integriert. Biopython bietet beispielsweise Möglichkeiten, Genbank Dateien zu parsen oder das Übersetzen von Nukleotidsequenzen in Aminosäuresequenzen anhand integrierter Übersetzungstabellen. [Brad Chapman und Jeffrey Chang, 2000].
Die Bibliotheken können unter `http://biopython.org/wiki/Download` heruntergeladen werden. Die aktuellste Version (1.62) wurde benutzt. (`http://biopython.org/DIST/biopython-1.62.zip`)

pySVG
pySVG[14] ist eine Python Bibliothek, die benutzt wird um SVG Dokumente bzw. Dateien zu erstellen. Die Bibliothek wurde von 2 Hobby-Programmierer in ihrer Freizeit erstellt und netterweise für die Öffentlichkeit zugänglich gemacht. Sie ist eigentlich eine Python Verpackung um SVG herum, die es dem Anwender erlaubt schnell und einfach SVG Elemente zu erstellen und in die Datei zu schreiben.
Es wurde ebenfalls die aktuellste Version (0.22) installiert, die jedoch nicht auf der Webseite verfügbar ist sonder unter `https://pypi.python.org/pypi/pysvg/0.2.2` heruntergeladen wurde.

sqlight3
sqlight3 ist normalerweise bei der Standardinstallation von Python schon enthalten, muss jedoch extra in den Programmcode implementiert werden. Mit Hilfe dieser Bibliothek ist es möglich SQlight Operationen auf SQlight Datenbanken durchzuführen.

[13] Die Module können entweder in einem Ordner des PYTHONPATH entpackt werden, oder mit PIP (`https://pypi.python.org/pypi/pip/`) installiert werden)
[14] http://codeboje.de/pysvg/

PIL
Die Python Imaging Library (PIL)[15] ist, wie der Name bereits sagt, eine Bibliothek um Bilder mit Python zu erstellen oder bearbeiten. Sie ist ebenfalls wie sqlight3 mit der Standardinstallation von Python mit integriert, muss aber ebenfalls extra in den Code eingebunden werden. Windows Benutzer müssen die Bibliothek extra herunterladen und dem PYTHONPATH hinzufügen.

3.1.3 SVG

Scalable Vector Graphics (SVG) ist eine auf XML basierendes Vektorgraphikformat für zweidimensionale Bilder, das Animationen und Interaktivität unterstützt. Die SVG Spezifikation ist ein offener Standard, der von dem W3C seit 1999 entwickelt wird. Bei dem Entstehungsprozess wurde zuerst versucht ein Format aus 6 Wettbewerbern (u.A. VML von Microsoft) zu wählen. Da aber keines den Anforderungen vollständig gerecht wurde, hat das W3C ein eigenes Format, das SVG, entwickelt. Dieses Format arbeitet hervorragend mit anderen W3C Standards, wie CSS oder DOM, zusammen. [W3C, 2010]

Die SVG Bilder, sowie ihr Verhalten, sind durch XML Textdateien definiert. Das hat zur Folge, dass sie Durchsucht, Indiziert, Gescriptet oder Komprimiert werden können. Wie bei XML Dateien, können sie mit einem Texteditor bearbeitet werden, es besteht aber auch die Möglichkeit Bilder mit einem Bildbearbeitungsprogramm wie Inkscape[16] zu erstellen. SVG wird inzwischen von allen namhaften Browsern (Firefox, Chrome, etc.) unterstützt und findet in einigen bekannten Tools (z.B. Google Maps) bereits Anwendung. [Mozilla Developer Network et al., 2013]

Das Dateiformat SVG wurde aus Folgenden Gründen gewählt:
Zum einen bringt es die Vorteile einer Vektorgraphik mit sich, die beispielsweise eine Verpixelung bei hoher Zoomstufe verhindert (siehe Abbildung 3.1). Zum Anderen erlaubt es ein Bild als Hintergrundbild zu deklarieren und danach Graphische Elemente darüber zu legen. Das hat den Vorteil, dass die Orginaldatei unberührt bleibt und beugt damit möglichen Problemen mit dem Copyright vor. Außerdem ist die SVG Datei klein und lässt sich gut verpacken, dass das Weiterleiten dieser vereinfacht und außerdem spart dies Ressourcen im Netzwerk und auf der Festplatte. Der Empfänger muss jedoch das Hintergrundbild auf seinem Computer haben, sowie den Pfad in der SVG Datei zu diesem ändern.
Aber es besteht der Nachteil, dass spezielle Programme benötigt werden um die SVG Dateien einfach zu betrachten. Die Browser zeigen zwar die Dateien an, das Zoomen oder Manövrieren ist jedoch nicht gut umgesetzt worden. Der Autor empfiehlt GIMP[17] für das Betrachten und Bearbeiten der SVG Dateien.

[15] http://pythonware.com/products/pil/
[16] `http://inkscape.org/`
[17] `http://www.gimp.org/`

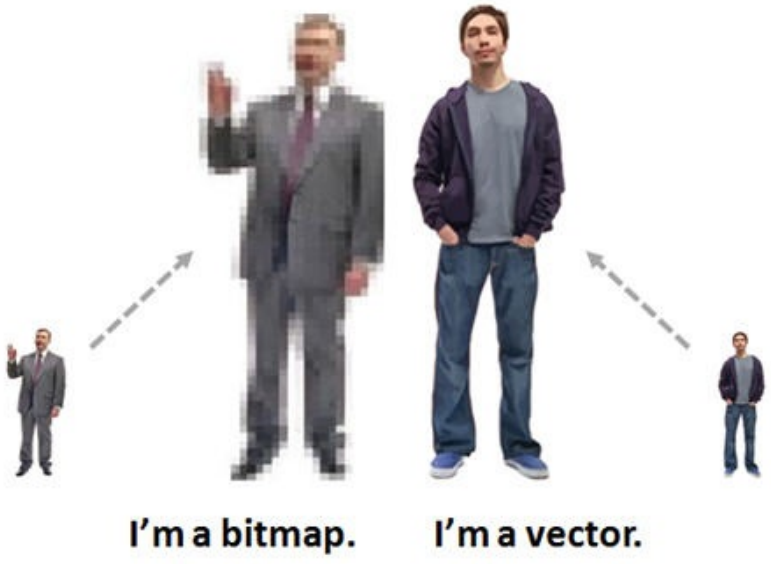

Abbildung 3.1: Vergleich von einer Bitmapgrafik (z.B. PNG) mit einer Vektorgraphik (z.B. SVG). [IMG-5]

3.1.4 PNG

Anfang der 1995er wurde bekannt, dass die LZW Kompressionsmethode, die in dem damals weit verbreitetem GIF Format Anwendung findet, von Unisys patentiert ist. Um keine Copyrightprobleme zu bekommen, und andern Einschränkungen des GIF Formats, wie die Beschränkung auf 256 Farben, zu entkommen, wurde das PNG (Portable Network Graphics) entworfen. Individuelle Programmierer von CompuServe, dem W3C und Weitere formierten eine Gruppe die das neue Format entwickelten und es offiziell 1996 Veröffentlichten. Dieses bekam auch gleich eine W3C Empfehlung, das einer Standarisierung gleich kommt. [Roelofs, 2013]

PNG wurde für das Internet entworfen, und ist somit nicht für Bilder mit hoher Qualitätsansprüchen zu empfehlen, da beispielsweise keine Einbindung von nicht-RGB Farbräumen (z.B. CMYK) vorhanden ist. Aber es unterstützt die 24-bit RGB oder die 32-bit RGBA Farbpalette, die eine Stufenweise einstellbare Transparenz ermöglicht. Außerdem wird die offene, verlustfreie Kompression DEFLATE[18] benutzt, die die Dateigröße eines kleinen Bitmap Bildes um bis zu 50% reduzieren kann. [Roelofs, 2013]

Um den Nachteilen von SVG entgegenzuwirken wurde auch das PNG Dateiformat in das Programm integriert. Dies soll ein schnelles und unkompliziertes Betrachten der Bilder auf jedem Rechner ermöglichen, da PNG mit jedem Bildbetrachtungsprogramm kompatibel ist. Es bleibt jedoch der Nachteil, dass jedes Bild neu erstellt wird und dies viel Speicher beansprucht.

[18] https://en.wikipedia.org/wiki/DEFLATE

3.1.5 SQLight

SQLite ist ein relationales Datenbankmanagementsystem, das nur eine kleine Bibliothek benötigt. Im Vergleich zu den SQL Datenbanken, benötigt es keine Konfiguration, keinen Server und besteht nur aus einer plattformunabhängigen Datei. Die Transaktionen sind ACID (Atomicity, Consistency, Isolation, Durability), selbst wenn das System währenddessen abstürzt. Der Syntax der Transaktionen ist dem Syntax von SQL sehr ähnlich, hat jedoch auch einige Eigenheiten. Dennoch ist ein Umstieg von SQL zu SQLight einfach. Das Projekt steht unter der Public Domain und wird deshalb von einer großen Community gepflegt, erweitert und ständig überprüft. [SQLite, 2013]

Das Datenbanksystem wurde Aufgrund der Plattformunabhängigkeit und des serverlosen Gebrauchs genommen. Außerdem ist die gestellte Datenbank, für die Koordinaten der Enzyme auf der Karte „Biochemical Pathways“, eine SQLight Datenbank. Um mögliche Portierungsprobleme zu beispielsweise PostgreSQL vorzubeugen wurde das System gleich so benutzt.

3.1.6 Genbank

Der Name Genbank steht zum einen für eine Sequenzdatenbank, sowie auch für das verwendete Dateiformat. Die Datenbank wird von dem NCBI (National Center for Biotechnology Information) als Teil der Kollaboration mit der EMBL (European Molecular Biology Laboratory) Datenbibliothek und der DNA Datenbank von Japan verwaltet. Sie beinhaltet alle öffentlichen Nukleotidsequenzen und ihre übersetzten Proteine. Die Daten werden von verschiedenen Instituten auf der Welt von über 100’000 verschiedenen Organismen per Sequenzierung gewonnen und eingesendet. Dort werden sie maschinell auf Qualität und Gültigkeit überprüft und können dann öffentlich über den FTP Server oder Entrez heruntergeladen werden. [Mizrachi, 2007]

Das Genbank-Dateiformat (siehe Abbildung 3.2) enthält genaue Informationen über die Quelle, eine eindeutige ID,die komplette Sequenz des Plasmids oder Chromosoms und die Position der Gene und ihre Übersetzung in die AS-Sequenz. Des Weiteren können noch zusätzliche Informationen, wie Jornual-Artikel die sich mit dem Eintrag beschäftigen, oder nähere Infos zu den Genen wie EC-Nummer, Name des Gens, Funktion, etc., eingebunden werden.

Das Format wurde ausgewählt, da es oft alle benötigten Informationen beinhaltet, wie die EC-Nummer, die Gensequenz, etc.
Die Genbankdateien wurden wie folgt heruntergeladen:

1. Über die DSMZ Webseite [19] wurde nach der DSM Nummer (z.B. 498 für *E.coli*

[19] `http://www.dsmz.de/catalogues/catalogue-microorganisms.html`

```
LOCUS       SCU49845     5028 bp    DNA     PLN       21-JUN-1999
DEFINITION  Saccharomyces cerevisiae TCP1-beta gene, partial cds,
    [...]
ACCESSION   U49845
VERSION     U49845.1  GI:1293613
KEYWORDS    .
SOURCE      Saccharomyces cerevisiae (baker's yeast)
  ORGANISM  Saccharomyces cerevisiae
            Eukaryota; Fungi; Ascomycota; Saccharomycotina; [...]
REFERENCE   1  (bases 1 to 5028)
  AUTHORS   Torpey,L.E., Gibbs,P.E., Nelson,J. and Lawrence,C.W.
  TITLE     Cloning and sequence of REV7, a gene whose function
     [...]
  JOURNAL   Yeast 10 (11), 1503-1509 (1994)
  PUBMED    7871890 [...]
FEATURES             Location/Qualifiers
     source          1..5028
                     /organism="Saccharomyces cerevisiae"
                     /db_xref="taxon:4932"
                     /chromosome="IX"
                     /map="9"
     CDS             <1..206
                     /codon_start=3
                     /product="TCP1-beta"
                     /protein_id="AAA98665.1"
                     /db_xref="GI:1293614"
                     /translation="SSIYN[...]"
     gene            687..3158
                     /gene="AXL2"
     CDS             687..3158
                     /gene="AXL2"
                     /note="plasma membrane glycoprotein"
                     /codon_start=1
                     /function="required for axial budding pattern
                        of S.
                     cerevisiae"
                     /product="Axl2p"
                     /protein_id="AAA98666.1"
                     /db_xref="GI:1293615"
                     /translation="MTQLQISLL[...]"

ORIGIN
        1 gatcctccat atacaacggt atctccacct caggtttaga tctcaacaac
          ggaaccattg
       61 ccgacatgag [...]
//
```

Abbildung 3.2: Eine verkürzte Beispiel-Genbank-Datei.

K12) des Organismus gesucht und der Treffer ausgewählt.
2. Der Name und Stamm wurde notiert und mit ihnen auf der KEGG Seite[20]nach Einträgen gesucht. Es mussten verschiedene Suchanfragen vorgenommen werden, da der exakte Stamm oder Name nicht gefunden wurde. Bei der Suche nach *Escherichia coli K-12* wurde der Treffer mit dem Stamm MG1655 ausgewählt.
3. Dort folgte man entweder direkt dem Link der NC-Nummer (NC_000913) oder suchte mit der Nummer auf der NCBI Webseite [21].
4. Der Genbank Eintrag (Display Settings -> Genbank Full), sowie die komplette Sequenz (Customize view -> Show sequence) muss komplett angezeigt werden.
5. Am Schluss wurde die Datei auf dem lokalen Computer gespeichert (Send-> Choose Destination -> File)

Auf diese Weise wurde das Genom von folgenden Organismen mit der DSM Nummer gewonnen.

Genome mit EC-Nummern, sowie Swiss-Prot und GI Nummer

- 498 - *Escherichia coli str K-12-substr mg1688*:
  NC_000913
- 3580 - *Zymomonas mobilis subsp pomaceae* und *ZM4 chromosome*:
  NC_015709,NC_006526
- 15444 - *Cupriavidus necator N1*, *Ralstonia eutropha H16* und *Ralstonia eutropha JMP134*:
  NC_015726, NC_015723, NC_015727, NC_015724, NC_008313,
  NC_008314, NC_005241, NC_007347, NC_007348, NC_007336, NC_007337

Genome mit Swiss-Prot und GI Nummer

- 20174 - *Lactobacillus plantarum WCFS1*:
  NC_004567

Genome nur mit GI Nummer

- 65 - *Paracoccus denitrificans*:
  NC_008686, NC_008687, NC_008688
- 3580 - *Zymomonas mobilis subsp pomaceae*:
  NC_015715, NC_015716
- 20174 - *Lactobacillus plantarum JDM1*, *Lactobacillus plantarum subsp plantarum ST-III* und *Lactobacillus plantarum WCFS1*:
  NC_012984, NC_014554, NC_014558, NC_006375, NC_006376,
  NC_006377

[20] `http://www.kegg.jp/`
[21] `http://www.ncbi.nlm.nih.gov/nuccore/`

- 30205 - *Agrobacterium radiobacter K84*:
  NC_011985, NC_011983, NC_011994, NC_011990, NC_011987

Da nur Proteine, die Enzyme kodieren für diese Arbeit wichtig sind, wurden nur die Genome mit EC-Nummern genommen. Einzelne der anderen Genome wurde untersucht, konnten jedoch nicht weiter ausgewertet werden, da keine EC-Nummern zu Proteinen in den Genbankdateien hinterlegt waren.

3.1.7 Benötigte Dateien

Das Programm wurde auf Basis der folgenden Dateien entwickelt.

Datensatz

Operation	Verbleibende Sequenzen	Verbleibend (%)
Ursprungsdatei	7168616	100%
Quality Trimmer	7165059	99,95%
Clipped	6253214	87,23%
Collapsed	2054554	28,66%
Ohne LSU Parc	1673827	23,35%
Ohne SSU Parc	1619337	22,59%
Ohne Refseq RNA	1588391	22,16%
Hits in nt	894455	12,48%

Abbildung 3.3: Auflistung der verbleibenden Sequenzen nach jedem Bearbeitungsschritt.

Der verwendete Datensatz wurde von Frau Stark erstellt. Dafür hat sie mit dem MasterPure RNA Purification Kit die RNA von folgenden Organismen extrahiert: *Paracoccus pantotrophus, Zymomonas mobilis, Rhizobium radiobacter, Lactobacillus plantarum und E.coli*. Der Organismus *Ralstonia eutropha* wurde mit der TRI-Reagent LS Methode bearbeitet. Die rRNA wurde mit dem mRNA ONLY Procaryotic mRNA Isolation Kit von der Firma Epicentre abgereichert. Die Einzelproben wurden gepoolt und zum Sequenzieren zur Firma GATC Biotech gesendet.

Der daraus entstandene Datensatz wurde von Herrn Kind mit den Folgenden Methoden verkürzt. (Abbildung 3.3) Mit dem FASTX-Toolkit wurde ein Qualtiy Trimming durchgeführt, die Sequenz-Primer entfernt, sowie Sequenzen, die kürzer als 20 Nukleotide gelöscht. Danach wurden identische Sequenzen zusammengefasst. Anschießend wurde die rRNA durch Abgleichen der Sequenzen gegen die LSU, SSU Parc und Refseq RNA

Datenbank entfernt. Am Schluss wurden die Sequenzen gegen die nicht-redundante Nukleotiddatenbank des NCBI annotiert.

Der daraus resultierende Datensatz besteht aus 1'603'675 Sequenzen und die am häufigsten Vorkommende ist mit einer Anzahl von 210'315 Reads „GATCGGAAGAGCACACGTCTGAACTCCAGTCACCGATGTATCTCGTATGCCGTCTTCTGCTTGAAAAAA - AAAA". Der Datensatz hat 1'080'328 Reads, die nur einmal vorkommen und die FASTA Datei hat eine Größe von 145'973 KB.

Genom
Die Genome wurden wie in Kapitel 3.1.6 beschrieben von den NCBI Servern heruntergeladen. Es wurde dabei darauf geachtet, dass die Dateien vollständig mit der „ORIGIN" Sequenz vorhanden sind.

SQLight Datenbanken
Die SQLight Datenbank mit den Koordinaten der EC Nummern auf der Karte „Biochemical Pathways" wurde von Prof. Wünschiers bereit gestellt. Die Datenbank, die für die Gruppierung der Treffer nach enzymatischen Ontologie genommen worden ist, wurde von Herrn Kind als PostgreSQL Datenbank bereitgestellt und von dem Autor in eine SQLight Datenbank portiert.

ENZYME und PROSITE
Die ENZYME nomenclatur database (enzyme.dat) wurde von EXPASY[22] heruntergeladen. Sie enthält eine Liste der EC-Nummern mit dazugehörigen Referenzen zu PROSITE oder Swiss-Prot. In dem PROSITE data file (PROSITE.dat) von der PROSITE[23] Webseite stehen alle Pattern und Profile mit ihren PRODoc IDs. Durch Verknüpfen der beiden Datenbanken anhand der PRODoc ID ist es möglich PROSITE Pattern EC-Nummern zuzuordnen.

Die Karte „Biochemical Pathways"
Als Stoffwechselkarte wurde eine PNG Bilddatei mit den Maßen 6871x4804 und einer Größe von 910KB verwendet. Sie enthält die „Biochemical Pathway" Karte in Graustufen.

3.2 Programmentwurf

Die Hauptaufgabe des Programmes ist das Suchen von Gensequenzen von bestimmten Organismen in einem bereitgestellten Datensatz und das Darstellen der Treffer auf „Biochemical Pathways". Dazu muss das Genbankfiles des Organismus eingelesen und die Gensequenzen mit dazugehöriger EC-Nummer extrahiert werden. Das Durchsuchen

[22] `ftp://ftp.expasy.org/databases/enzyme/`
[23] `ftp://ftp.expasy.org/databases/PROSITE/`

des Datensatzes nach den Gensequenzen wird Aufgrund der Schnelligkeit, und der erhöhten Signifikanz durch das Hinzunehmen von ähnlichen Sequenzen, BLAST verwendet. Dazu muss das Genom in eine Datenbank umgewandelt werden und danach die einzelnen Sequenzen des Datensatzes gegen diese geblasted werden. Die Ausgabe von BLAST wird analysiert und den den EC-Nummern der Treffer Koordinaten aus einer Datenbank zugeordnet. Am Ende soll ein Bild gezeichnet werden, das die Treffer mit ihrer Anzahl an Funden auf der Stoffwechselkarte hervorhebt.

Um einen Vergleich von verschiedenen Organismen zu bekommen, benötigt das Programm eine Funktion um den oben genannten Durchlauf mehrmals durchzuführen, jedoch mit unterschiedlichen Genomen. Anschließend sollen die einzelnen Ergebnisse miteinander Verglichen werden und dies sowohl tabellarisch, als auch auf der Böhringer Map.

Das Auffinden von Gensequenzen in dem Datensatz hängt von der Vollständigkeit und Korrektheit der Genbank Dateien ab. Da dies nicht immer gegeben ist, wurde dem Programm noch eine PROSITE Suche hinzugefügt. Dabei wird der Datensatz mit PROSITE Pattern durchsucht, denen vorher EC-Nummern zugeordnet worden sind. Danach werden wie bei dem BLAST-Lauf die Ergebnisse Analysiert, Gruppiert und Visualisiert.

Bei der Erstellung des Programmes wurde vor allem darauf geachtet, dass wenig Arbeitsspeicher benötigt wird. Dies wurde durch ein sofortiges Schreiben der Daten auf die Festplatte realisiert. Der Nachteil dabei ist, dass dadurch das Programm verlangsamt wird, da HDD Zugriffe viel Zeit (im Vergleich zu RAM Zugriffe) in Anspruch nehmen. Auf der anderen Seite jedoch sorgt das Schreiben der Zwischenergebnisse in Textdateien für Sicherheit. So gehen beispielsweise die Daten bei einem Systemabsturz nicht verloren. Des Weiteren kann so jeder Schritt nachvollzogen werden, was bei der Fehlersuche hilfreich ist. Und Schließlich bringt diese Form des Sicherns die Möglichkeit verschiedene Berechnungen an verschiedenen PCs ausführen zu lassen, und danach die Dateien zusammen zu führen und eine abschließende Berechnung an einem Computer zu vollziehen. Dies würde die Gesamtberechnungszeit eines Multi-BLAST Laufes deutlich verkürzen.

Eine weitere Besonderheit ist die Trennung der einzelnen Aufgaben in Module. Jedes Modul kümmert sich dabei um einen Teil des Programmes. Dies hat den Vorteil, dass zum Einen einzelne Module in anderen Programmen verwendet werden können, ohne das ganze Programm zu implementieren. Zum Anderen vereinfacht dies das Erweitern und Verbessern des Programmes, da dann nur einzelne Module ausgetauscht werden müssen.

3.3 Programmaufbau

3.3.1 Module

Das Programm wurde in 8 Module unterteilt (siehe Abbildung 3.4 und Abbildung 3.5):

- Mapper:
  Das Hauptmodul Mapper koordiniert das ganze Programm. Es enthält die Main-Methode, die mit dem Start des Programmes aufgerufen wird. Je nachdem welche Parameter dem Programm übergeben werden, führt die Methode die entsprechenden Arbeitsschritte aus und benutzt dabei alle weiteren Module. Die Klasse c_Logger dient dem Festhalten der Konsolenausgabe in einer Textdatei, die in das Output-Verzeichnis am Ende des Programmes kopiert wird.
- m_ClassData:
  Dieses Modul enthält die Klasse c_ClassData. Eine Instanz von ihr wird am Anfang des Programmes kreiert und das Objekt wird dann von Methode zu Methode weiter-, bzw. zurückgegeben. In dem Objekt werden alle benötigten Daten hinterlegt, wie die Pfadnamen zu den einzelnen Dateien, die Namen der zu erstellenden Dateien oder der E-Wert. Um die Informationen abzugreifen wurden entsprechende Getter und Setter Methoden geschrieben, die gleichzeitig die Gültigkeit der Eingabe überprüfen. So wird bei der Eingabe des Pfades zu dem BLAST-Ordner überprüft, ob die Programme „BLASTn“ und „makeblastdb“ in diesem Verzeichnis vorhanden sind. Die Methode „reloadExtraPaths“ setzt alle zusätzlichen Pfad- und Dateinamen, die nicht vom Benutzer eingegeben werden müssen.
- m_Input:
  Mit Hilfe von m_Input wird entweder eine Datei eingelesen, oder per Eingabedialog vom Benutzer alle benötigten Informationen dem Programm übergeben.
- m_SQLight:
  Für den Zugriff auf die SQLight Datenbanken wurde dieses Modul erstellt. Die Methoden geben die Koordinaten zu den EC-Nummern, oder die Pathway-ID zu diesen zurück.
- m_Blast:
  M_Blast ist das größte der 8 Module. Mit den Methoden lassen sich unter Anderem eine Genbank Datei in eine Fasta Datei umwandeln, eine Datenbank dieser erstellen um den Datensatz dagegen blasten zu lassen. Des Weiteren wird die Blastausgabe analysiert und dabei die Treffer mit den EC-Nummern extrahiert. Diese werden Gruppiert und dann Koordinaten zugewiesen. Der Ablauf ist der gleiche für einen MultiBLASTrun mit mehreren Organismen, mit der Besonderheit, dass die einzelnen Schritte zum Teil ihre eigenen Methoden benötigen.
- m_Prosite
  So wie m_Blast alles für einen BLAST Vergleich enthält, so bietet m_Prosite alle Methoden die für eine komplette Überprüfung des Datensatzes mit PROSITE Pat-

tern an. Die Ergebnisse nach einer PROSITE Suche werden so umgeschrieben, dass sie am Ende wie die BLAST Ergebnisse aussehen und mit den gleichen Paint Methoden weiter verarbeitet werden können.

- m_PNG und m_SVG:
  Die beiden Module Zeichnen aus den Ergebnissen die Stoffwechselkarte in den Formaten PNG und SVG. Das Modul m_PNG benötigt eine Times New Roman TrueTypeFont Datei in dem „Files“ Order des Programmes.

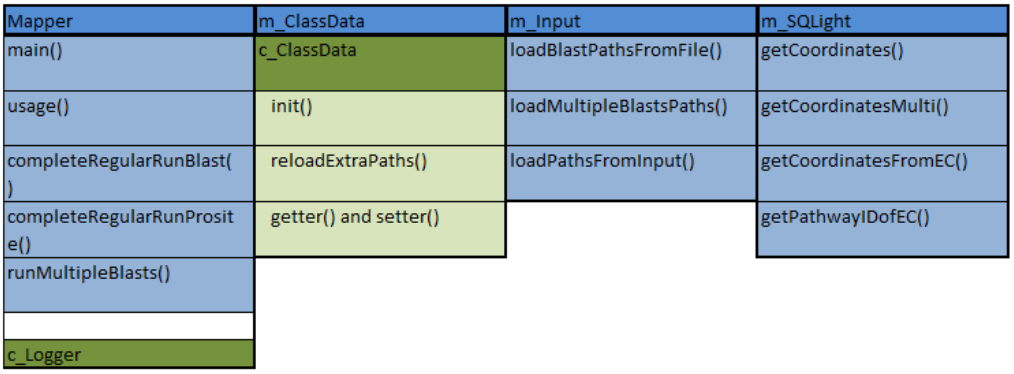

Abbildung 3.4: Moduldiagramm 1 der verwendeten Module. Blau zeigt Module und deren Methoden an, grün Klassen und deren Methoden.

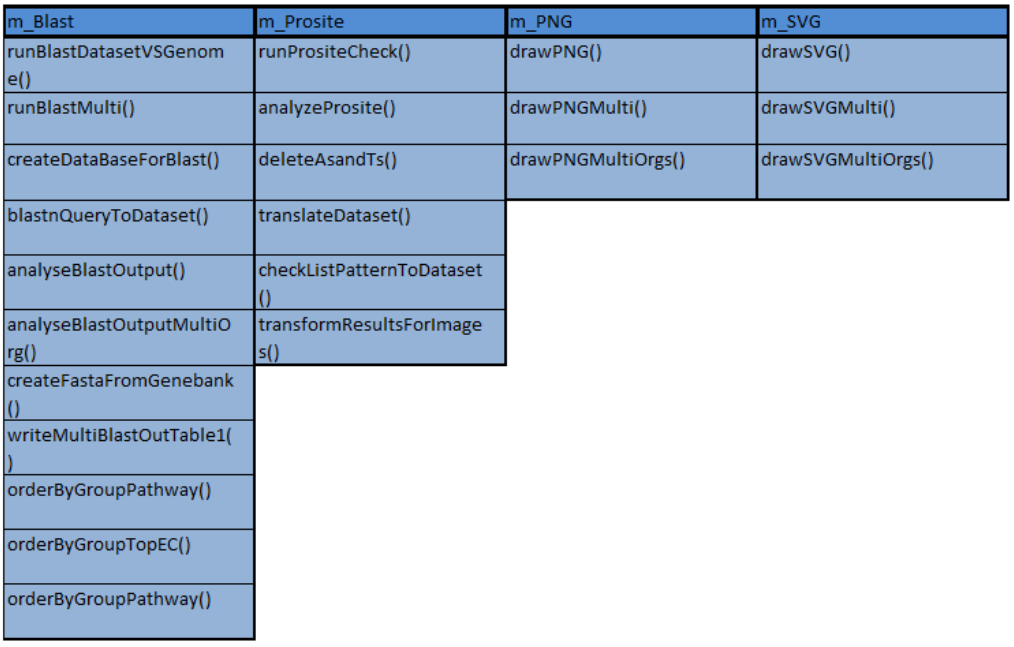

Abbildung 3.5: Moduldiagramm 2/2 der verwendeten Module.

3.3.2 Input

Die Eingabe der Daten kann auf unterschiedliche Weise erfolgen. Die Pfade, Namen und Werte können entweder als Parameter, als Datei oder per Eingabedialog übergeben werden.

Parameter

Zum einen können diese als Parameter bei dem Programmstart übergeben werden. Die einzelnen Kommandos sind wie folgt:

- **-h** oder **--help**: Aufruf der Hilfe
- **-f** oder **--file** mit Argument: Einlesen der Daten von der als Argument übergebenen Datei. (z.B. -f /usr/share/file.txt)
- **-i** oder **--inputdialog**: Startet die Eingabe der Daten mit einem Eingabedialog.
- **-r** oder **--run** mit Argument: Die als Argument übergebene Zeichen geben den Ablauf des Programmes wieder. (z.B. -r r)
  Dabei steht „r“ für einen kompletten BLAST Lauf, „p“ für einen kompletten PROSITE Lauf, „m“ für einen Multi-BLAST Lauf. Des Weiteren kann jeder einzelne Ablauf extra aufgerufen werden. Für nähere Informationen dazu bitte die Hilfsdatei des Programmes aufrufen.
- **-o** oder **--output** mit Argument: Das Argument gibt den Ausgabeordner wieder. Ist dieser nicht vorhanden wird er erstellt. (z.B. -o /media/Data/Output)
- **-n** oder **--name** mit Argument: Das Argument gibt den Namen des Projektes wieder. Es wird ein Ordner mit diesem Namen in dem Ausgabeordner erstellt und alle erstellten Dateien werden dort gespeichert. (z.B. -n PROSITERun)
- **-d** oder **--dataset** mit Argument: Das Argument gibt den Pfad und Name des Datasets wieder. Es muss eine FASTA-Datei sein. (z.B. -d /media/Data/dataset.fasta)
- **-g** oder **--genome** mit Argument: Das Argument gibt den Pfad und Name des Genom wieder. Es muss eine Genbank-Datei sein. (z.B. -g /media/Data/genome.gb)
- **-m** oder **--map** mit Argument: Das Argument gibt den Pfad und Name der Bilddatei der Stoffwechselkarte wieder. Es muss eine PNG Datei mit den Maßen 6871x4804 sein. (z.B. -m /media/Data/boepath.png)
- **-c** oder **--coordinates** mit Argument: Das Argument gibt den Pfad und Name der Datenbankdatei für die Koordinaten der EC-Nummern zu der Böhringer Map wieder. Es muss eine SQLight Datenbank-Datei sein. (z.B. -c /media/map_db.db)
- **-b** oder **--BLAST** mit Argument: Das Argument gibt den Pfad zu dem Ordner wieder, in dem sich die BLAST Programme befinden. Die Programme BLASTn und makeBLASTdb in diesem Ordner vorhanden sein. (z.B. -b /media/ncbi-BLAST-2.2.28+/bin)
- **-e** oder **--evalue** mit Argument: Das Argument legt den E-Wert fest, mit dem das BLAST Programm arbeitet. Es muss ein ganzzahliger Wert über 0 sein. (z.B. -e

1)

- **-s** oder **--scan** mit Argument: Das Argument gibt den Pfad und Name des PROSITE Programmes wieder. Es muss das Perl Script sein, das von der PROSITE Webseite zur Verfügung gestellt wird.(z.B. -s /media/ps_scan/ps_scan.pl)
- **-p** oder **--PROSITE** mit Argument: Das Argument gibt den Pfad und Name der PROSITE.dat Datei wieder. Es muss die von der PROSITE Webseite zur Verfügung gestellte Datei sein. (z.B. -p /media/Data/PROSITE.dat)
- **-z** oder **--enzyme** mit Argument: Das Argument gibt den Pfad und Name der enzyme.dat Datei wieder. Es muss die von Expasy zur Verfügung gestellte Datei sein. (z.B. -e /media/Data/enzyme.dat)
- **-g** oder **--grouping** mit Argument: Das Argument gibt den Pfad und Name der Datenbankdatei für die Gruppierung der Treffer nach Ontologie wieder. Es muss die vom Autor erstellte SQLight Datenbank-Datei sein. (z.B. -c /media/Data/grouping_db.sqlight3)

Ein Beispielaufruf für einen BLAST Lauf wäre:
„python Mapper.py -f /usr/share/input.txt -r r“

Aufbau der Eingabedateien

Bei der Übergabe der Daten von einer Datei für einen BLAST Lauf muss diese in der folgenden Reihenfolge aufgebaut sein. Eine Beispieldatei ist in dem „Files“ Verzeichnis zu finden:

- Pfad des Ausgabeordners
- Name des Projekts
- Pfad der BLAST-Programme
- E-Value
- Name und Pfad der Stoffwechselkarte
- Name und Pfad der Koordinaten-Datenbank-Datei
- Name und Pfad der Gruppierung-Datenbank-Datei
- Name und Pfad des Datasets
- Name und Pfad des Genoms

Wenn ein Multi-BLAST Lauf durchgeführt werden soll, bleibt die Reihenfolge wie oben bestehen, jedoch wird der Name des Projekt hinter einem Tabstop nach dem Namen und Pfad des Genoms geschrieben. (z.B. /media/Data/genome.gb genome1). Die einzelnen Genome der verschiedenen Organismen mit ihrem Projektnamen werden hintereinander in jeweils einer neuen Zeile geschrieben.

Eingabe der Daten per Eingabedialog

Wenn der Parameter „-i“ mit Übergeben wird, startet ein Konsolenbasierter Eingabedialog. Der Benutzer wird immer nach dem Pfad oder Wert der jeweiligen Datei gefragt und dieser muss ihn dann Eingeben.

3.3.3 Output

Das Programm erstellt in seinem Durchlauf viele Dateien. Die wichtigsten werden nun erläutert:

- BLAST+Output+<Name Query>—<Name Datenbank>.txt:
  Das ist die direkte Ausgabe des Blastdurchlaufes. Sie enthält die Treffer mit den Informationen in tabellarischer Form. Das erste Feld gibt alle Informationen zu der Gensequenz, wie EC-Nummer, Proteinname oder die GI Nummer, jeweils mit einem „+“ voneinander getrennt. Das nächste Feld gibt den Header eines Datensatzes-Reads an. Darauf folgt der E-Wert, die Prozentuale Übereinstimmung, die Alignment-Länge und die Prozentuale positive Übereinstimmung.

```
6.3.2.2+ybdk+UniProtKB/Swiss-Prot:P77213+GI:16128564    8387-67 6e-11   100.00  30      100.00
6.3.2.2+ybdk+UniProtKB/Swiss-Prot:P77213+GI:16128564    8551-66 2e-06   100.00  22      100.00
6.3.2.2+ybdk+UniProtKB/Swiss-Prot:P77213+GI:16128564    8641-65 1e-40   100.00  81      100.00
6.3.2.2+ybdk+UniProtKB/Swiss-Prot:P77213+GI:16128564    8732-64 1e-52   100.00  101     100.00
No-EC-Number+grol+UniProtKB/Swiss-Prot:P0A6F5+GI:16131968       8736-64 2e-05   86.00   50      86.00
No-EC-Number+grol+UniProtKB/Swiss-Prot:P0A6F5+GI:16131968       9273-61 2e-08   82.29   96      82.29
6.3.2.2+ybdk+UniProtKB/Swiss-Prot:P77213+GI:16128564    9932-57 6e-32   100.00  66      100.00
```

Abbildung 3.6: Ausschnitt der BLAST+Output Datei.

Bei der geparseten Ausgabedatei sind nur noch die für das Erstellen des Bildes benötigten Dateien enthalten: Die EC-Nummer und die Anzahl der Sequenzen, die mit dem Collapser (von Gabriel) zusammengefasst worden sind.

```
3.4.24.-        2
No-EC-Number    2
2.7.7.6 2
3.4.21.53       2
No-EC-Number    2
No-EC-Number    2
```

Abbildung 3.7: Ausschnitt der geparseten BLAST+Output Datei.

- BLAST+EC_Numbers_Valid+Summed.txt:
  Gibt die gültigen EC-Nummern mit der Anzahl der Treffer wieder.

```
1.1.1.154	2
1.1.1.158	5
1.1.1.159	19
1.1.1.17	4
1.1.1.193	2
1.1.1.205	11
1.1.1.215	2
1.1.1.22	182
```

Abbildung 3.8: Ausschnitt der BLAST+EC_Numbers_Valid+Summed Datei.

- Matches_Final+Ordered_by_<top_EC oder Pathway>:
  Bei diesen beiden Dateien wurden die Treffer entweder nach dem „Top Level Code“ oder nach der Ontologie zusammengefasst. Die top_EC Datei gibt den Namen der Gruppe, die Ziffer, die Anzahl der BLAST-Treffer und die tatsächliche Anzahl der Treffer in dem Datensatz (die Anzahl der Zusammengefassten Sequenzen wurde mit einbezogen) wieder.
  Die Pathway Datei beinhaltet die Ontologie-ID, den Namen, sowie die Anzahl der BLAST Treffer und die tatsächlichen Treffer.

```
map00051	Fructose and mannose metabolism	9	163
map00052	Galactose metabolism	8	212
map00053	Ascorbate and aldarate metabolism	5	214
map00061	Fatty acid biosynthesis	5	190
map00062	Fatty acid elongation	3	76
map00071	Fatty acid metabolism	6	736
```

Abbildung 3.9: Ausschnitt der Matches_Final+Ordered_by_Pathway Datei.

```
Oxidoreductases	1.*.*.*	49	2091
Transferases	2.*.*.*	98	5457
Hydrolases	3.*.*.*	27	446
Lyases	4.*.*.*	43	2327
Isomerases	5.*.*.*	18	296
Ligases	6.*.*.*	26	7796
```

Abbildung 3.10: Ausschnitt der Matches_Final+Ordered_by_top_EC Datei.

- Matches_Final+Sort_by_<Hits oder EC>:
  Diese Datei wird als letztes Erstellt und auf dieser Basiert das Zeichnen des Bildes. Sie besteht aus der EC-Nummer, die Anzahl der Treffer und den Koordinaten zu der jeweiligen EC-Nummer. Es wurde dabei zum Einen nach der EC-Nummer sotiert, zum Anderen nach der Anzahl der Treffer.

```
2.4.1.1	85	[[258, 2356, 55, 38], [149, 2388, 65, 32]]
5.4.2.1	88	[[2130, 2330, 54, 28], [2115, 2379, 86, 31]]
2.2.1.2	89	[[1485, 2221, 79, 10]]
3.2.1.22	91	[[675, 1510, 96, 13]]
4.1.1.23	99	[[4992, 3680, 166, 11]]
6.1.1.9	100	[[865, 4454, 55, 19]]
4.2.1.9	107	[[526, 4351, 85, 20], [558, 4140, 85, 21]]
6.3.1.2	113	[[3898, 2947, 54, 28]]
4.3.1.1	128	[[3379, 1933, 85, 19]]
```

Abbildung 3.11: Ausschnitt der Matches_Final+Sort_by_Hits Datei.

- Image_Hits_(PNG oder SVG):
  Am Ende eines vollständigen Durchlaufes wird ein Bild als PNG und SVG Datei gezeichnet. Dieses hat als Hintergrund die Stoffwechselkarte. Die Legende in der oberen linken Ecke des Bildes gibt eine Beschreibung der Farben, sowie noch ein paar Zusatzinformationen, wie den Maximal- oder Minimalwert, wieder.

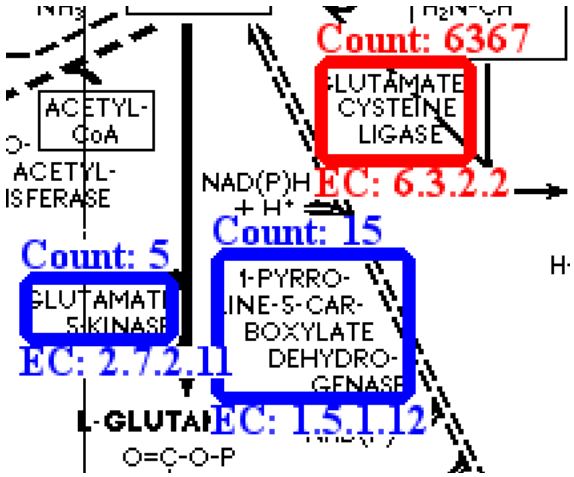

Abbildung 3.12: Ausschnitt aus einer der Bilddateien.

  Bei dem Zeichnen eines Multi-BLAST-Bildes ist jedem Organismus eine Farbe zugeordnet. Wegen der Übersichtlichkeit werden maximal nur 10 Organismen dargestellt. Die Größe des Balkens gibt von oben nach unten die Anzahl der Treffer wieder. Dabei sind die Balken im Verhältnis zu der höchsten Anzahl an Treffern des Enzyms berechnet. Dieser Wert steht über dem Balkendiagramm. (Siehe Bilder Kapitel 5.5)
- BLAST+Multi_Output+Table_<Zahl>.txt:
  Zur besseren Darstellung und Weiterverarbeitung wurden die Ergebnisse des Multi-BLAST Laufes in verschiedenen tabellarischen Formen abgespeichert. Die Dateien können mit einem Tabellenkalkulationsprogramm wie Excel importiert werden.
  Die Form Table_1 ist Zeilenweise aufgebaut. Jede Zeile beinhaltet alle Gene und Organismen, die zu einem Dataset-Read gefunden worden sind. Die Zeile beginnt mit dem Header, danach der Name des 1. Organismus mit Doppelpunkt, gefolgt von den Genen mit den jeweiligen Geninformationen durch ein Semikolon getrennt, dann der Name des 2. Organismus mit den Genen usw.

```
3387-67»Genome_(498)_Escherichia_coli_str_K-12-substr_mg1688: 6.3.2.2;ybdk;P77213
3551-66»Genome_(498)_Escherichia_coli_str_K-12-substr_mg1688: 6.3.2.2;ybdk;P77213
3641-65»Genome_(498)_Escherichia_coli_str_K-12-substr_mg1688: 6.3.2.2;ybdk;P77213
3732-64»Genome_(498)_Escherichia_coli_str_K-12-substr_mg1688: 6.3.2.2;ybdk;P77213
3736-64»Genome_(498)_Escherichia_coli_str_K-12-substr_mg1688: No-EC-Number;grol;P0A6F5»
```

Abbildung 3.13: Ausschnitt aus der Tabelle 1 - Datei.

Table_2 ist Spaltenweiseweise aufgebaut. In Spalte 1 steht der Header, dann der Organismus, die EC-Nummer, der Genname und die UniProtKB/Swiss-Prot Nummer.

19989-27	Genome_(498)_Escherichia_coli_str_K-12-substr_mg1688	6.3.2.2	ybdk	P77213
20424-27	Genome_(498)_Escherichia_coli_str_K-12-substr_mg1688	6.3.2.2	ybdk	P77213
20438-27	Genome_(498)_Escherichia_coli_str_K-12-substr_mg1688	No-EC-Number	grol	P0A6F5
	Genome_(3580)_Zymomonas_mobilis_ZM4_chromosome	No-EC-Number	groel	IPR002423
	Genome_(3580)_Zymomonas_mobilis_subsp_pomaceae	No-EC-Number	No-GeneName	IPR002423
	Genome_(15444)_Ralstonia_eutropha_H16_C_1	No-EC-Number	groel	113866733
20446-27	Genome_(498)_Escherichia_coli_str_K-12-substr_mg1688	No-EC-Number	dnak	P0A6Y8
	Genome_(3580)_Zymomonas_mobilis_ZM4_chromosome	No-EC-Number	dnak	IPR002048
	Genome_(3580)_Zymomonas_mobilis_subsp_pomaceae	No-EC-Number	No-GeneName	IPR012725

Tabelle 3.1: Ausschnitt aus der Tabelle 2 - Datei, die in LibreOffice Calc importiert wurde.

3.3.4 Gültigkeit

Der Autor der Bachelorarbeit und Programmierer des Programmes hat nur Basiskenntnisse in der Mathematik. Folglich kann er nicht mathematisch die Gültigkeit der einzelnen Algorithmen überprüfen. Jedoch wurde ein kurzes Testdataset, ein kleines Testgenom, etc. erstellt und das Programm mit diesen Daten laufen gelassen. Anschließend wurden der Versuch per Hand nachgerechnet und mit den Ergebnissen des Programmes verglichen. Es wurden sogar Nukleotidsequenzen erstellt, die in Aminosäuren übersetzt den Namen des Autors wiedergeben um die Methode für das Übersetzen des Datensatzes zu testen. Bei diesen Versuchen wurden die meisten Grenzfälle, wie Lücken in den Sequenzen, Punktmutationen, etc. mit abgedeckt. Falls die Ergebnisse voneinander abgewichen sind, wurde der Code so lange verbessert, bis eine 100%ige Übereinstimmung vorlag. Somit sollte das Programm wie gewünscht arbeiten und die Ergebnisse Aussagekräftig sein.

3.4 Programmablauf

Das Programm lässt sich in drei Unterprogramme aufteilen: Die BLAST Suche , die PROSITE Suche und in einen Multi-BLAST Lauf mit jeweils der Analyse der Ergebnisse.

3.4.1 BLAST Suche

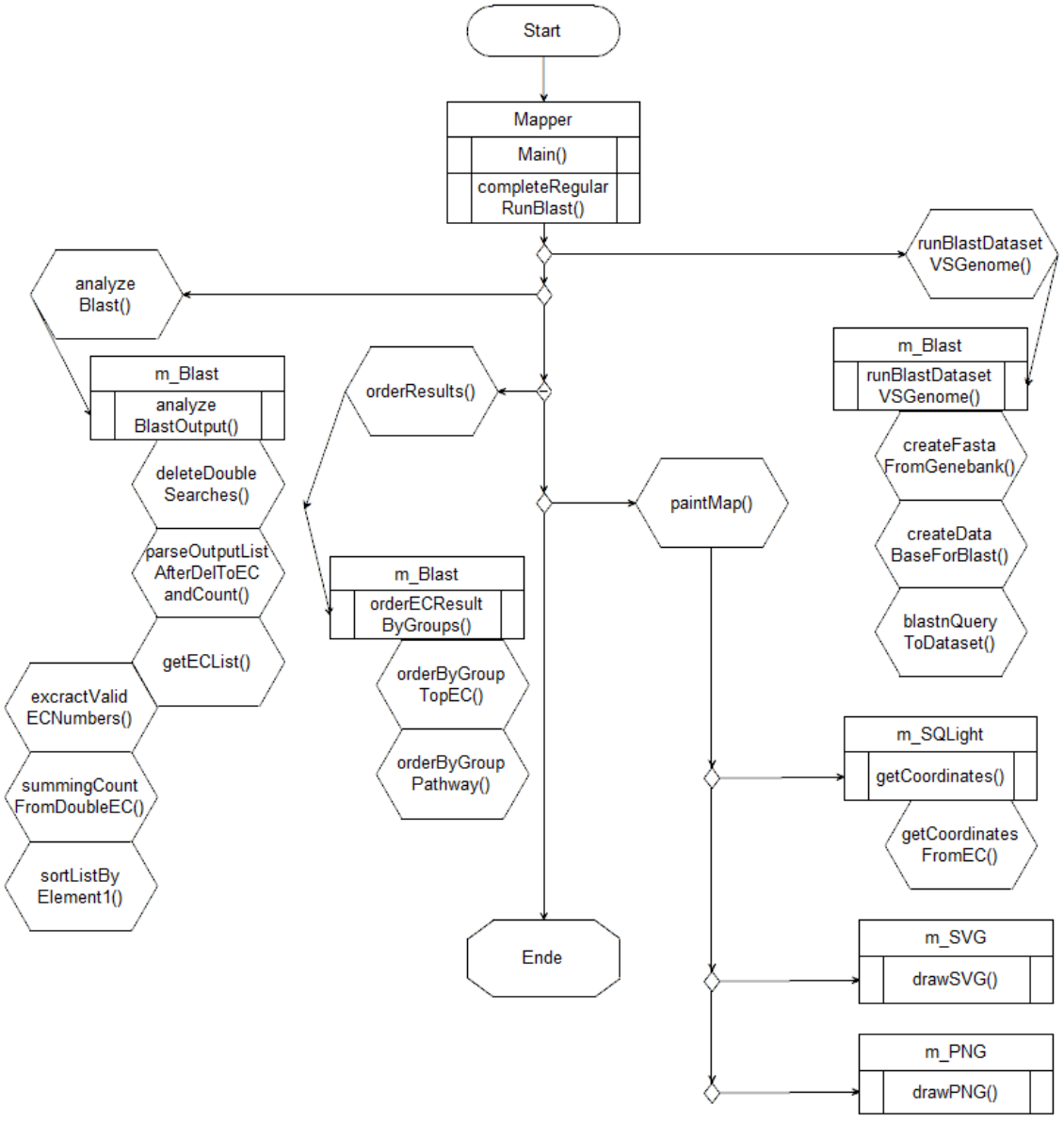

Abbildung 3.14: Flussdiagramm für eine BLAST Suche.

Mit der Eingabe der Option „r“ mit dem Parameter „-r“ beim Starten des Programmes wird ein normaler BLAST Lauf mit Analyse und abschließendem Zeichnen des Bildes durchgeführt. Am Anfang wird die Methode completeRegularRunBLAST() aufgerufen, die dann die jeweiligen „Untermethoden“ aufruft.
Als Erstes wird über die Methode runBLASTDatasetVSGenome() die Methode runBLASTDatasetVSGenome() des Moduls m_BLAST aufgerufen. Diese kreiert mit createFastaFromGenebank() eine Fasta-Datei aus einer Genbank-Datei. Unter Zuhilfenahme von Biopyhton Modulen wird die Genbank-Datei nach Einträgen mit dem Feature „CDS“ gesucht und dann die Sequenz und andere Geninformationen, wie EC-Nummer extra-

hiert. Falls die Information nicht vorhanden ist, steht bei dem Eintrag dann beispielsweise „No-EC-Number„. Die Geninformationen bilden dann den Header für die Sequenz des Gens, wobei jede Information mit einem Plus-Zeichen voneinander getrennt ist. Mit der Methode createDataBaseForBLAST() wird eine Datenbank aus einer Fasta-Datei mit dem Aufruf des Programmes „makeBLASTdb“ erzeugt. Danach wird der Datensatz gegen die Datenbank mit der Methode BLASTnQueryToDataset. Dabei wird das Programm „BLASTn“ mit den Parametern “-db <Name Datenbank> -query <Name des Datasets> -evalue <E-Wert> -word_size 7 -penalty -3 -reward 1 -out <Name der Output-Datei> -outfmt 6 sseqid qseqid evalue pident length ppos“ aufgerufen.
Im nächsten Schritt wird mit der Methode analyzeBLAST() die Methode analyszeBLASTOutput() des Moduls m_BLAST ausgeführt. Diese löscht alle doppelten Einträge der BLAST Suche mit deleteDoubleSearches(). In seltenen Fällen kann es vorkommen, dass bei dem Vergleich der gleichen Sequenzen zwei oder mehr Treffer auftauchen, jedoch mit unterschiedlicher Alignmentlänge. Da diese das Ergebnis verfälschen würden, müssen alle zusätzlichen Funde gelöscht werden. Nach dem Löschen wird der BLAST-Output in ein Listenformat geparsed, das die Weiterverarbeitung erleichtert. Am Ende der Analyse wird aus der geparten BLAST-Ausgabe nur die gültigen EC-Nummern mit der Anzahl der Treffer extrahiert, und diese dann zusammengefasst. Bei manchen EC-Nummern fehlt beispielsweise die letze Ziffer, oder es gibt keine EC-Nummer. Die Herausgefilterten Treffer werden zur Sicherheit in eine Datei geschrieben.
Nun werden die Treffer mit den Methoden orderByGroupTopEC() und orderByGroupPathway() in den Methoden orderECResultByGroups() vom Modul m_BLAST und orderResults() vom Modul m_Mapper sortiert. Mit orderByGroupTopEC() werden einfach alle Einträge nach der ersten Ziffer (Top Level Code) der EC-Nummer zusammengefasst. OrderByGroupPathway() vergleicht die EC-Nummern mit den Einträgen der Gruppierungs-Datenbank und fasst die gleiche Ontologien zusammen.
Mit der letzten Methode paintMap() werden die Koordinaten den EC-Nummern mit der Methode getCoordinates() vom Modul m_SQLight zugeordnet. Dabei wird die Koordinaten-Datenbank mit SQLight befehlen Aufgerufen. Anschließend wird das SVG Bild mit drawSVG() vom Modul m_SVG und das PNG Bild mit drawPNG() vom Modul m_PNG gezeichnet. DrawSVG() nimmt Elemente von pysvg und drawPNG() von PIL zu Hilfe. Damit bei dem Erstellen der SVG Datei das Hintergrundbild hinzugefügt wird, ruft die Methode am Ende addImage() auf, die die nötigen Zeilen Code in die Datei schreibt.

3.4.2 Multi-BLAST Lauf

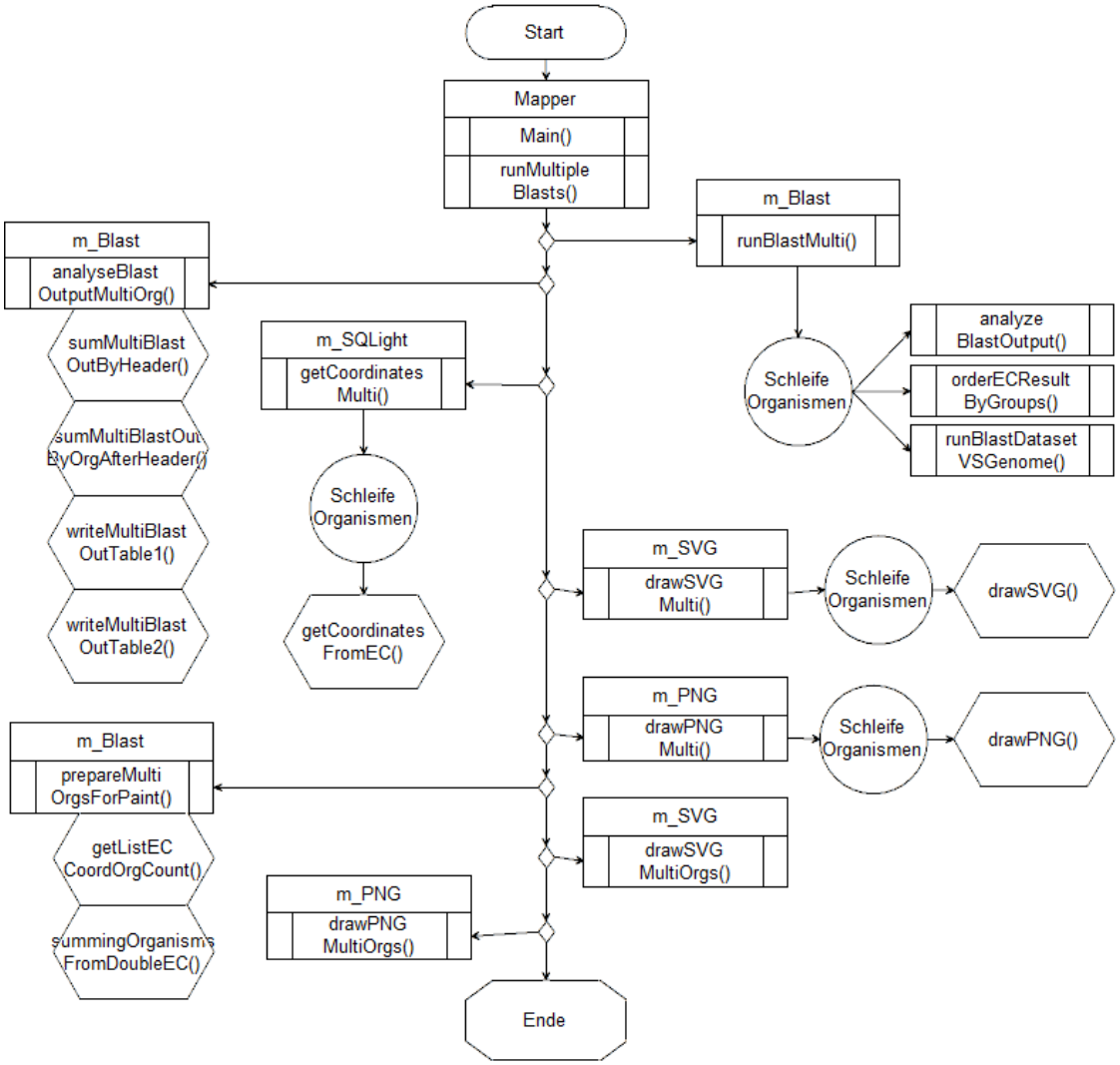

Abbildung 3.15: Flussdiagramm für einen Multi-BLAST Lauf.

Der Multi-BLAST Lauf ähnelt der normalen BLAST Suche. Am Anfang wird mit der Methode runBLASTMulti() des Moduls m_BLAST über eine Schleife der Organismen wie oben erwähnt eine Fasta-Datei aus der Genbank-Datei erstellt, mit dieser eine Datenbank erstellt und danach der Datensatz gegen die Datenbank geBLASTet. Jede einzelne Ausgabe wird Analysiert und Sortiert, jedoch noch nicht Gezeichnet. Danach werden alle Ausgaben in eine Datei geschrieben, jedoch vorher mit dem Organismus getagged. Die Methode analyseBLASTOutputMultiOrg() untersucht diese Datei weiter, indem sie mit sumMultiBLASTOutByHeader() zuerst alle Einträge mit dem gleichen Header zusammenfasst. Anschließend wird mit sumMultiBLASTOutByOrgAfterHeader() die Einträge mit den gleichen Organismen (unter dem Header) zusammengetragen. Dadurch wird eine mehrdimensionale Liste erzeugt, die als erstes Feld den Header hat und als

zweites Feld eine Liste mit den Organismen, die in ihrem Genom eine Übereinstimmung mit der Sequenz unter dem Header haben. Jeder Organismus besteht wiederum aus einer Liste bei der das erste Feld der Name des Organismus ist und das zweite eine Liste der Gene, die mit der Headersequenz übereinstimmen. Zu diesen Genen sind alle Informationen vorhanden, die bei dem Erstellen der Fasta-Datei extrahiert worden sind. Mit dieser komplexen Liste werden die beiden Tabellen in ihrer jeweiligen Form geschrieben. WriteMultiBLASTOutTable1 schreibt die Informationen Zeilenweise nach Header sortiert in eine Datei, writeMultiBLASTOutTable2 schreibt diese Spaltenweise nach den jeweiligen Feldern Sortiert. Nach diesem Schritt wird mit der Methode getCoordinatesMulti() von m_SQLight von jedem Organismus einzeln die Koordinaten mit den EC-Nummern verknüpft. Dafür läuft eine Schleife über die Organismen ab, die mit jedem Durchlauf getCoordinates() (s. oben) aufruft. Daraufhin wird nach dem gleichen Schema die Bilder mit drawSVGMulti() von m_SVG und drawPNGMulti() von m_PNG gezeichnet. Nun werden die unter getCoordinatesMulti() erstellten einzelnen Dateien mit ihrem jeweiligen Organismus getagged und mit getListECCoordOrgCount() zusammengetragen und mit summingOrganismsFromDoubleEC() nach den EC-Nummern sotiert. Zu jeder EC-Nummer steht nur eine Liste mit den Organismen und Anzahl der Treffern zur Verfügung. Mit dieser Liste wird abschließend das Bild mit drawSVGMultiOrgs() von m_SVG und drawPNGMultiOrgs() von m_PNG gemalt.

3.4.3 PROSITE Suche

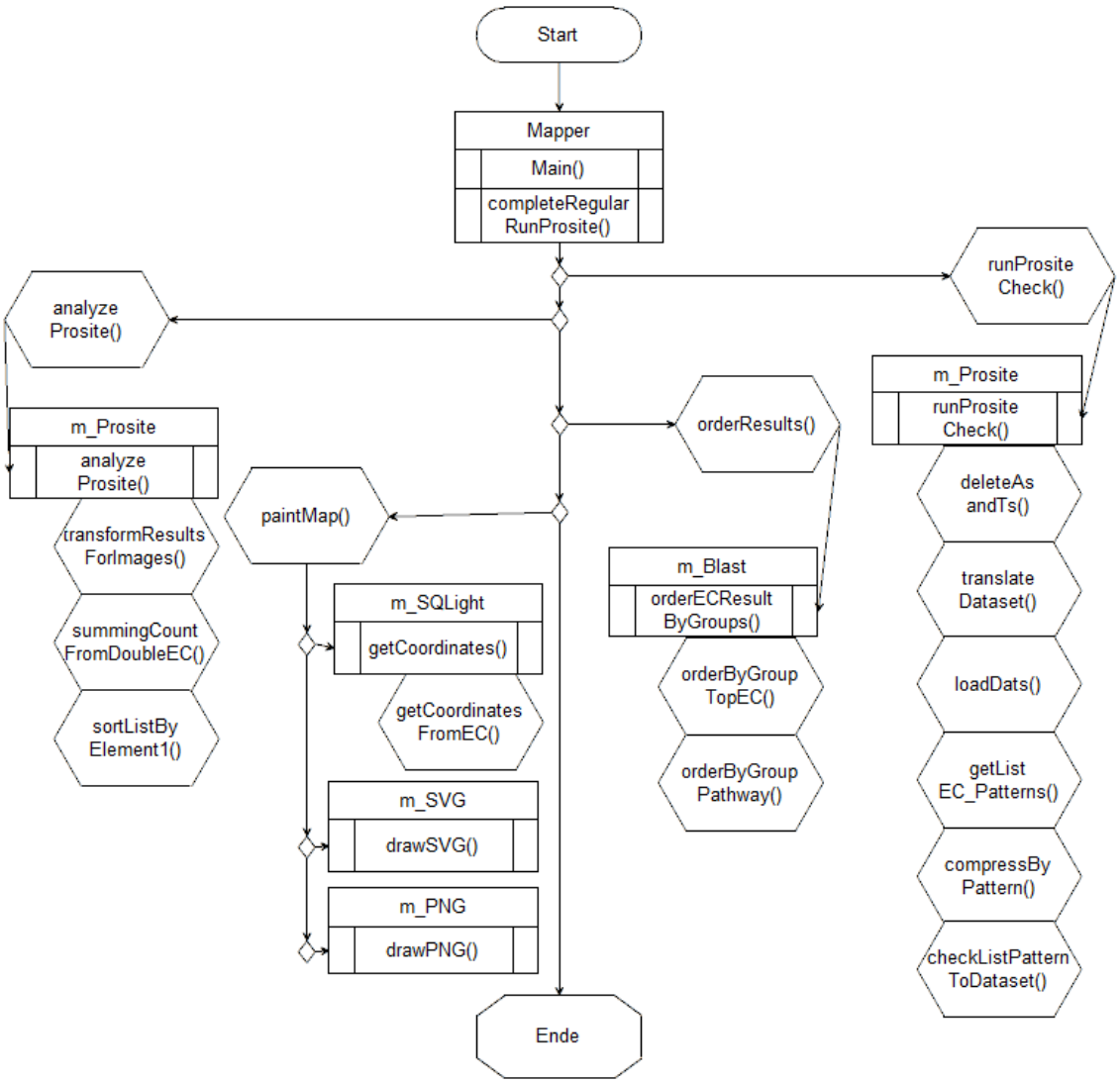

Abbildung 3.16: Flussdiagramm für eine PROSITE Suche.

Die Methode completeRegularRunPROSITE() startet die PROSITE Suche mit der „Untermethode“ runPROSITECheck(), die daraufhin von dem Modul m_PROSITE runPROSITECheck() aufruft. Zunächst werden alle Einträge mit 10 oder mehr hintereinanderliegenden Nukleotiden A oder T aus dem Datensatz gelöscht. Diese sind nicht aussagekräft und verbrauchen nur unnötigen Speicher und Rechenleistung. Anschließend werden die Nukleotidsequenzen des Datensatzes mit translateDataset() in die 6 möglichen Proteinsequenzen mit der Übersetzungstabelle 11 (Bacterial and Plant Plastid) übersetzt. Anschließend werden die Dateien Enzyme.dat und PROSITE.dat mit loadDats() in zwei Listen geladen. Diese werden mit getListEC_Patterns() über die PDOC ID miteinander Verknüpft, so dass jedem PROSITE Pattern die EC-Nummern zugeordnet werden. Die Einträge werden mit compressByPattern() nach den Pattern zusammen-

gefasst. Die daraus resultierende Liste an einmaligen PROSTIE Pattern wird für das Durchsuchen des Datensatzes nach Treffern mit der Methode checkListPatternToDataset() genommen. Die Analyse des Ergebnisses wird von der Methode analyzePROSITE() von m_PROSITE in der gleichnamigen Methode vom Modul Mapper übernommen. Dafür wird die Ausgabedatei mit transformResultsForImages() in die Listenform geparsed, die auch in der BLAST Suche nach dem Aufruf der Methode excractValidECNumbers() erstellt wird. Dadurch sind alle darauf Folgenden Schritte, wie das Summieren der Treffer mit der gleichen EC-Nummer, mit dem BLAST Lauf identisch, jedoch befinden sich alle Methoden in dem Modul m_PROSITE.

4 Durchführung

Um die Aufgabe, das Zuordnen der Reads zu Genen bestimmter Organismen, mit einfachen Mitteln zu erfüllen, wurde zuerst ein identischer Vergleich durchgeführt. Dabei wurden die Sequenzen der Reads mit den Sequenzen der Gene durch Buchstabe für Buchstabe überprüft. Da dieses Vorgehen schlechte Ergebnisse lieferte (siehe Kapitel 5.1 ist zügig eine Ähnlichkeitssuche implementiert und diese Funktion aus dem Programm genommen worden.
Die BLAST Suche lieferte deutlich bessere Ergebnisse ab und wurde somit weiter entwickelt. Zunächst wurde von dem Datensatz eine BLAST Datenbank erstellt und das Genom dagegen geblastet. Nun stellte sich die Frage was passiert, wenn der Ablauf gedreht wird. Also ist eine BLAST Datenbank von dem Genom erstellt worden und der Datensatz wurde damit verglichen. Dafür wurde das *E. coli* Genom und der E-Wert 1 genommen.
Nach diesem Test wurden BLAST Läufe mit verschiedenen E-Werten und Organismen durchgeführt. Es wurden Genome der unter Kapitel 3.1.6 unter „mit EC-Nummer“ beschriebenen Genbankfiles verwendet, da die anderen Genome keine Angaben zu den Enzymen beinhalteten und folglich nicht auf die Stoffwechselkarte gezeichnet werden konnten. Um diesen Vorgang zu Automatisierten ist der Multi-BLAST geschrieben worden. Mit ihm wurden die verschiedenen Genome mit einem E-Wert von 0.0001 gegen den Datensatz geBLASTet und sind danach miteinander verglichen worden. Dabei wurde zum Einen jedes einzelne Genom genommen, zum Anderen wurden die Genome des gleichen Organismus nach dem Umwandeln in einer FASTA-Datei zusammengefasst. Z.B. sind die Genome NC_008313 (Chromosom 1), NC_008314 (Chromosom 2) und NC_005241 (Megaplasmid pHG1) des Organismus *Cupriavidus necator H16* (DSM 15444) zu einem Gesamtgenom zusammengefasst worden. Es wurde ein Multi-BLAST Lauf mit den vereinigten Genomen von *E. coli K12* (498), *Zmobilis Pomaceae* (3580), und *R. Eutropha H16* (15444) durchgeführt.
Da die Anzahl der gefundenen EC-Nummern von der Vollständigkeit der Genbank-Dateien abhängt, wurde versucht den Datensatz mit Gensequenzen von jeder einzelnen EC-Nummer zu durchsuchen. Da eine solche Datenbank jedoch nicht exisitert, wurde ein Umweg ausgearbeitet. Es gibt eine Datenbank mit allen EC-Nummern, die PDOC IDs zu diesen enthalten. Und es ist eine Datenbank mit PROSITE Pattern vorhanden, die ebenfalls die dazugehörigen PDOC IDs beinhalten. Folglich wurden diese beiden Datenbanken zusammengeführt und der Datensatz mit den PROSTIE Pattern abgeglichen.

5 Ergebnisse und Diskussion

5.1 Identischer Vergleich

Bei der ersten Betrachtung der Bilder und Daten, ist schnell zu erkennen, dass der Identische Sequenzvergleich zu wenig Treffer bringt. Dies kommt von der Variabilität der natürlichen Sequenzen. In der Natur ist es normal, dass Punktmutuationen in den Sequenzen auftreten und dass sich das Genom von Organismus zu Organismus unterscheidet. Wenn also ein Nukleotid von der maximal 102 nt langen Readsequenzen gegenüber den Genomsequenzen abweicht und der Rest übereinstimmt, wird es dennoch nicht gefunden. Mit *E. coli* wurden die meisten identischen Übereinstimmungen gefunden. Bei den anderen Organismen war das Ergebnis noch schlechter. Es konnten 41 EC-Nummern von dem *E. coli* Genom dem Datensatzzugeordnet werden, *Zymomonas mobilis* nur 6 und *Cupriavidus necator* 21. Durch einen Vergleich der Ergebnisse mit denen einer BLAST Suche mit einem E-Wert von 1, wird die Variabilität der Natur deutlich gezeigt. Beispielsweise liefert die EC-Nummer 6.3.2.2 von dem *E. coli* Genom bei dem identischen Vergleich 245 Treffer, bei dem BLAST Lauf jedoch 6367 Treffer. Bei anderen Enzymen und Organismen ist der Unterschied ebenso stark vorhanden.
Aufgrund dieser schlechten Ergebnisse wurde wie oben erwähnt der identische Sequenzvergleich nicht weiter verfolgt und aus dem Programm genommen. Dieser Versuch zeigte lediglich erneut die Vielfalt der Natur und dass flexible Programme benötigt werden, um diese zu Erforschen.

5.2 BLAST Genome auf Datensatz Vs Datensatz auf Genom

Auf den ersten Blick sehen die Ergebnisse der beiden Durchläufe gleich aus, sowohl in den Bildern, als auch in den erstellten Tabellen bzw. Textdateien. Durch einen exakten Vergleich der Daten mit dem Linux Tool sdiff[24] wurde der erste Verdacht bestätigt. Es macht keinen Unterschied ob eine Datenbank von dem Datensatz erstellt wird, und das Genom dagegen geblastet wird, oder umgekehrt, wenn von dem Genom eine Datenbank kreiert wird und der Datensatz dagegen geblastet wird. Da das BLAST Alignment auf Basis des Smith–Waterman Algorithmus, der ein lokales Alignment durchführt, basiert, war das Ergebnis zu erwarten. Diese Erkenntnis hilft bei der Verbesserung des Programmes. Bisher wurde von dem Genom eine Datenbank erstellt. Folglich wird bei einem Multi-BLAST Lauf mit mehreren Organismen und Genomen viele Datenbanken erstellt. Es würde jedoch reichen, wenn nur von dem Datensatz eine Datenbank erstellt wird und gegen diese die einzelnen Genome geblastet werden. In der nächsten

[24] `http://linux.about.com/library/cmd/blcmdl1_sdiff.htm`

Überarbeitung des Programmes wird die Methode dementsprechen angepasst, dass sie weniger Speicher und Zeit durch das Erstellen der einzelnen Datenbanken, in Anspruch nimmt.

5.3 Beispiel E.coli

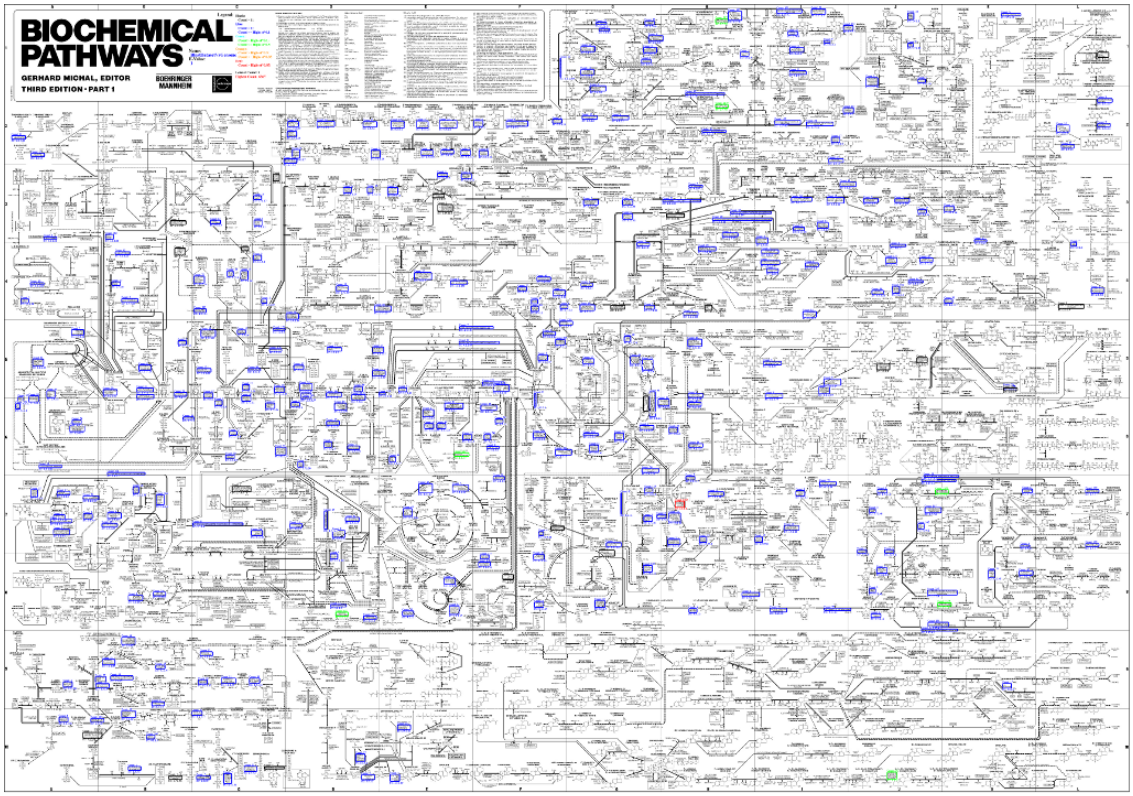

Abbildung 5.1: Das von dem Programm erstellte Bild mit dem E.coli Genom. Die Farben spiegeln die Häufigkeit der Expression im Vergleich zu der maximalen Anzahl an Treffern wieder. Rot bedeutet höchste Expression, Orange mäßig hohe Expression, Grün mittlere Expression, Blau geringe Expression und schwarz, dass nur eine Übereinstimmung mit dem Datensatz gefunden wurde.

Über *E. coli* sind die meisten Informationen erhalten und hat folglich eines der vollständigsten und am besten dokumentierten Genome. Durch die hervorragende Datenlage des Organismus wurde dieser genauer Untersucht und alle Versuche zunächst an ihm erpropt und danach auf andere Organismen erweitert.
Das am häufigsten gefundene Enzym ist bei einem BLAST Lauf mit einem E-Wert von 1 die (Gamma-) Glutamate-cysteine Ligase mit der EC-Nummer 6.3.2.2 und einer Trefferanzahl von 6367 (37% aller Treffer). Warum dieses Enzym eine so hohe Expression aufweißt ist unklar. Beim Menschen deutet dies auf ein bösartiges Melanom hin [Mougiakakos et al, 2012], jedoch wurde die RNA von Bakterien gewonnen. Das Enzym

hat weder ein lange noch kurze Aminosäurensequenz, hat keinen ungewöhnliche 3D-Struktur oder sonst ein ausergewöhnliches Merkmal. Als Ursache wird eine ungewöhnlich hohe Vervielfältigung des Stranges bei der Sequenzierung vermutet.
Bei dem gleichen Durchlauf kommen unter den 261 gefundenen Enzymen, 21 nur 1x in dem Datenstz vor, 233 unter 100x und nur 2 Enzyme über 1000x. Diese Anzahl erscheint recht gering bei den ca. 1,6 Millionen Reads in dem Datensatz. Drei Hauptursachen könnten diesen kleinen Anteil erklären. Erstens kann das aktuelle Genom der Organismen von dem Genom in den Genbank-Dateien abweichen. Zweitens kodieren nicht alle mRNAs Enzyme, das mit einem kurzen Blick in die Output Dateien bestätigt wird. Es gibt viele Treffer, die keine gültige EC-Nummer besitzten und deshalb aussortiert werden. Drittens sind die Reads maximal 102 nt Lang, so dass Gene über mehrere Reads verteilt Vorliegen. Da aber nur ein Read mit der Gensequenz verglichen wird, kann ein großes Stück fehlen und aufgrund der Unvollständigen Sequenz findet das BLAST Programm keine Vor. Eine deutliche Erhöhung des E-Wertes könnte eine Verbesserung verschaffen, jedoch erhöht dies auch die Wahrscheinlichkeit für falschpositive Treffer. Des Weiteren verstärkt dieser Schritt ein schon bereits bestehendes Problem. Wenn die Gensequenz über mehrere Reads Verteilt vorliegt und der BLAST Algorithmus erkennt eine Übereinstimmung der Reads mit der Sequenz, addiert das Mapper Programm fälschlicherweise die Anzahl der Treffer. Ein kurzes Beispiel zur Verdeutlichung, die Sequenz ZAA liegt 3x im Datensatz vor, die Sequenz BBB 2x und CCD 1x. Die Gensequenz lautet AABBBCC. BLAST würde eine Übereinstimmung der drei Sequenzen mit der Gensequenz finden. Das Mapper Programm gibt an, dass das Gen 6x in dem Datensatz gefunden wurde. Jedoch kann das Gen maximal 3x Exprimiert worden sein. Das Fehlen der anderen 2 Sequenzen kann natürlicher Ursprung, beispielsweise RNA Verdau sein. Dieser Fehler wurde bisher durch einen klein gehaltenen E-Wert reduziert, jedoch bringt dies den oben erwähnten Nachteil. Das Programm kann auf das Herausfiltern dieses Fehlers erweitert werden, oder die Sequenzen werden Assembleld und man erhält vollständige Zusammenhängende Sequenzen in dem Datensatz gegen diese dann mit einer höheren Erfolgsaussicht geBLASTet werden kann. Der Vollständigkeit halber wird noch erwähnt, dass das Sequenzieren keine exakte Wissenschaft ist und bei diesem Schritt auch Fehler auftauchen. Diese sind jedoch sehr gering so dass sie in dieser Größenordnung vernachlässigbar sind.
Das Ordnen der Ergebnisse nach dem Top Level Code bringt hervor, dass die meisten Treffer im Datensatz mit 7796 bei den Ligasen zu finden sind. Das Ergebnis lässt sich mit dem hohen Vorkommens der Glutamate-cystein Ligase begründen. Am wenigsten Vertreten sind die Isomerasen mit 296 Treffern. Dies könnte zum einen an der geringen Anzahl verschiedener Isomerasen liegen, oder an der weniger wichtigen und eher einfachen Aufgabe eine Verbindung in eine isomere Struktur zu katalysieren. Wenig Verwunderlich erscheint das Ergebnis der Sortierung nach Ontologie. 6592 von 17377 Treffer gehören dem Glutathione Stoffwechsel. Insgesamt wurden 103 verschiedene Stoffwechsel abgedeckt.

Mit anderen E-Werten unterscheiden sich natürlich die Ergebnisse, da durch eine Er-

höhung des Schwellwertes die Ungenauigkeit zunimmt, und somit die Signifikanz abnimmt. Dies hat den Effekt, dass mehr Enzyme in dem Datensatz gefunden werden, jedoch sinkt die Aussagekraft, da sich die Wahrscheinlichkeit erhöht, falsch-positive Enzyme zu finden. Bei der Erniedrigung des E-Values gilt folglich das Gegenteil. Da das Hauptaugenmerk zunächst auf das Finden von Enzymen in dem Datensatz lag, wurde häufig der etwas hohe E-Wert von 1 genommen.
Eine Forschergruppe hat ein online Tool[25] zur Verfügung gestellt, dass das Genom eines Organismus auf der Stoffwechselkarte hervorhebt. Wenn diese Karte mit der von dem Mapper Programm Erstellten übereinander gelegt werden (siehe Abbildung 5.2), ist eine große Übereinstimmung zu erkennen. Jedoch zeigt sich, dass Mapper mehr Gene des *E. coli* Genoms der Biochemical Pathways Karte zuordnen kann als G-Language. Vor allem im oberen Bereich der Karte, der sich mit der Verwendung von Guanosine befasst, findet das Programm Treffer, die dem Genome-Viewer entgangen sind. Der Vergleich bestätigt die Gültigkeit des Programmes, und zeigt, dass die meisten Enzyme von *E. coli* in dem Datensatz vorhanden sind.

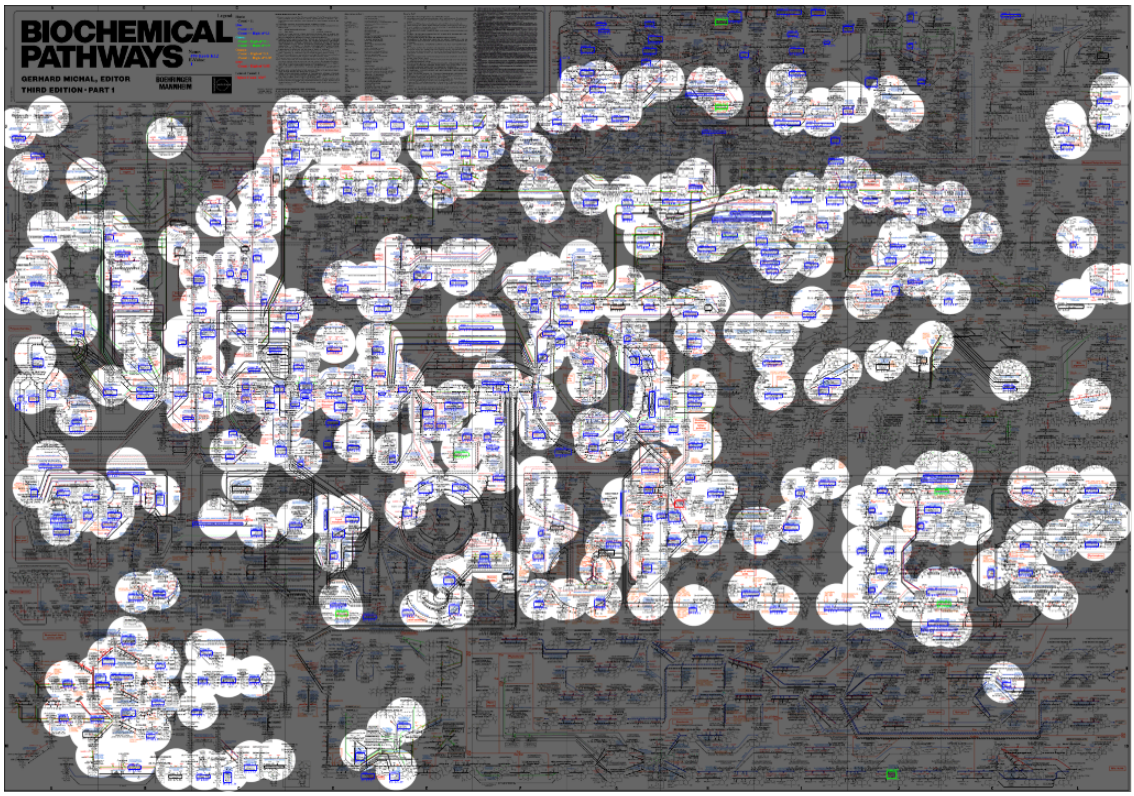

Abbildung 5.2: Übereinanderlagerung des Bildes von G3 und von dem Programm für E. coli.

[25] http://www.g-language.org/g3/

5.4 PROSITE-Lauf

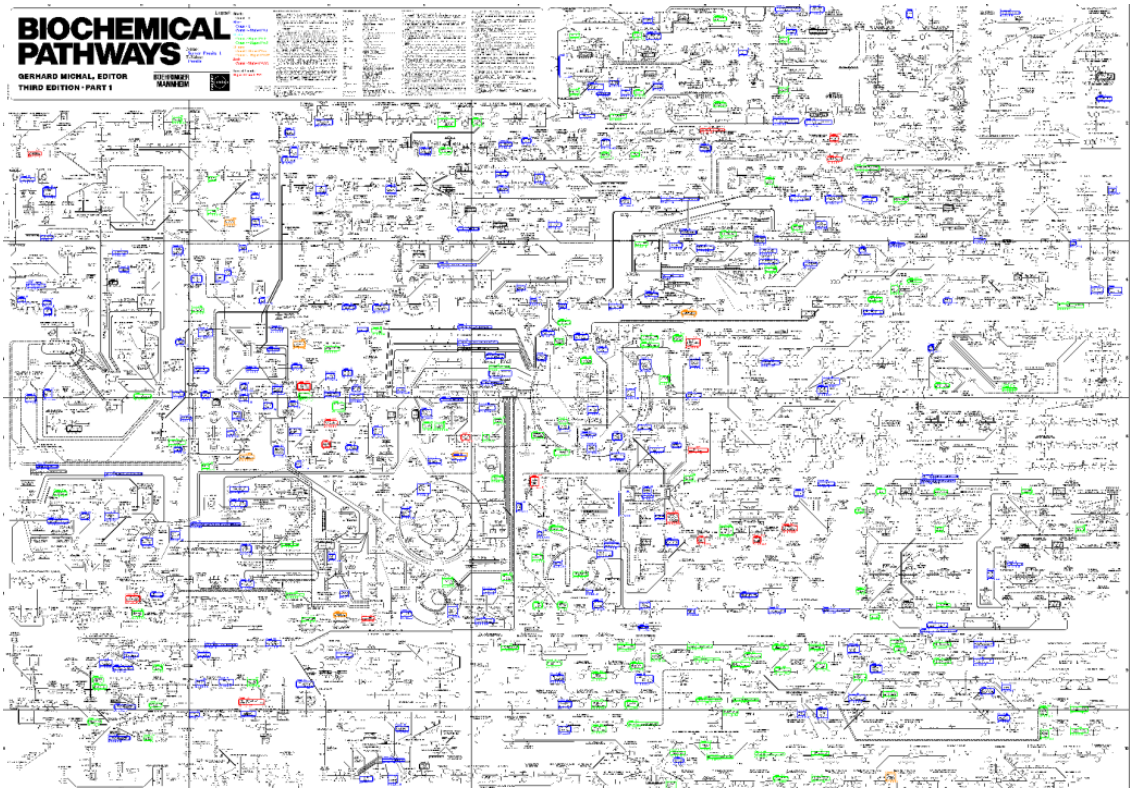

Abbildung 5.3: Erstellte Stoffwechselkarte nach einer PROSITE Suche.

Das Durchsuchen des Datensatzes mit PROSITE Pattern brachte 269 gefundene Enzyme zum Vorschein. Am häufigsten mit 537 Treffern sind die Aldehyde Dehydrogenasen mit der PDOC ID PDOC00068 gefunden worden. Diese 8 Enzyme teilen sich das gleiche PROSITE Pattern und haben deshalb die gleiche Häufigkeit. Lediglich 19 Enzyme hatten nur einen Treffer in dem Datensatz, 223 unter 100 Treffer und 250 unter 200 Treffern. Diese Verteilung unterscheidet sich deutlich von der eines BLAST Laufes. Dies war zu erwarten, da die Pattern deutlich kürzer sind, weil sie nur die konservierten Regionen beinhalten und nicht die komplette Gensequenz. Aus diesem Grund fallen sie dem oben erwähnten Problem mit der Aufteilung der Gensequenz über mehrere Reads, weniger zum Opfer fallen. Warum jedoch nur 269 Enzyme der 18898 möglichen Enzyme der Datanbank gefunden werden ist unklar. Es wurde zur Sicherheit ein erneuter Versuch unternommen, jedoch mit dem gleichen Ergebnis. Eine mögliche Erklärung ist die geringe Expression bestimmter Enzyme und dass die mRNA dieser schnell verdaut wird. Außerdem kodieren die Organismen nicht alle Enzyme der Datenbank, da diese auch Enzyme von anderen Spezien und Rassen enthält. Die genaue Anzahl der Enzyme in den verwendeten Organismen ist noch nicht komplett erforscht. Das *E. coli* Genom hat beispielsweise 1’110 Einträge mit einer EC-Nummer. Des Weiteren existiert nicht zu jedem Enzym bereits ein Pattern. Dies zeigt sich auch in dem Unterschied der gefunde-

nen Enzyme. Das häufigste Enzym (6.3.2.2) bei dem BLAST Lauf taucht beispielsweise in den Ergebnissen des PROSITE Laufes überhaupt nicht auf. Gleiches gilt für andere Enzyme.
Dennoch erweist sich die PROSITE Suche als Hilfreich, da Enzyme in dem Datensatz gefunden worden sind, die keinem der eingesetzten Organismen zuzuordnen waren. Dies liegt höchstwahrscheinlich jedoch an der Unvollständigkeit der Genbank-Dateien der verwendeten Stämme. Zu dem Organismus *Paracoccus pantotrophus* mit der DSM Nummer 65 liegt beispielsweise noch kein vollständig sequenziertes Genom vor. Folglich könnten die unbekannten Enzyme zu diesem Stamm gehören.
Wenn das Programm einen Datensatz von komplett unbekannten Organismen bearbeitet, werden durch den PROSITE Lauf dennoch Einblicke in den Metabolismus der Organismen gewonnen, aber mit der Einschränkung, dass diese nicht vollständig sind.

5.5 MultiBLAST

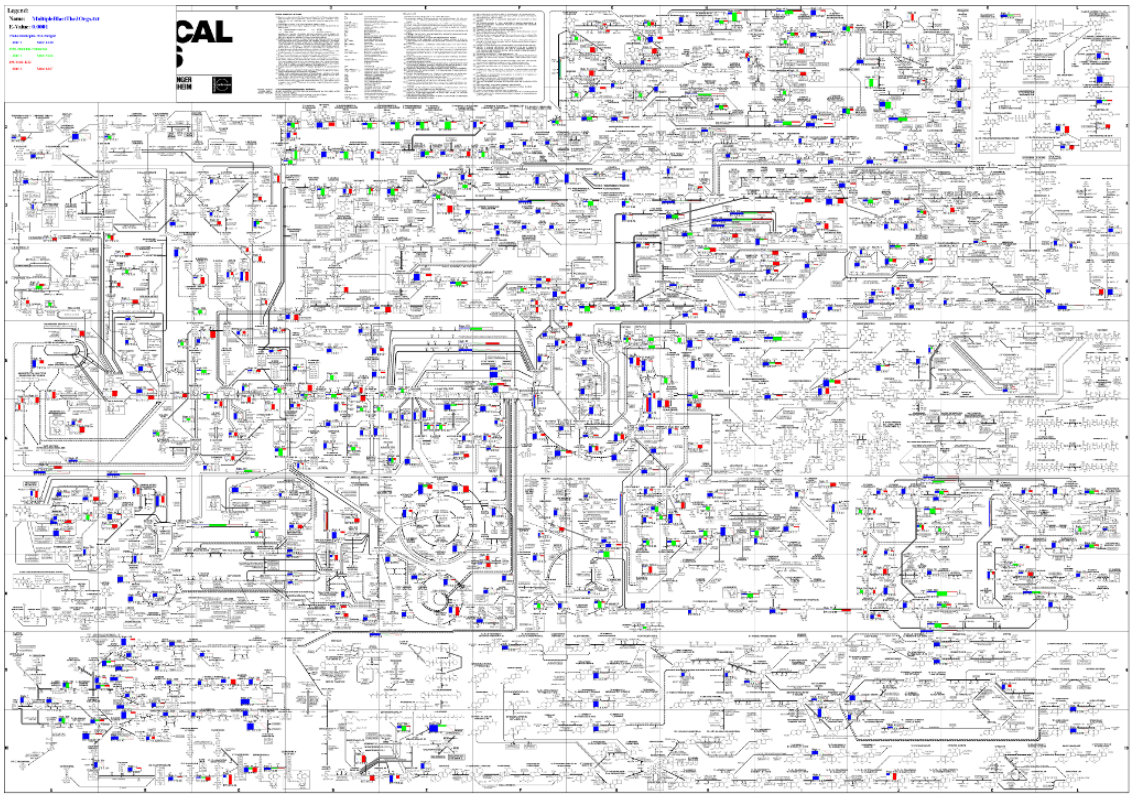

Abbildung 5.4: Erstellte Stoffwechselkarte bei einem Vergleich der Zusammengefassten Genome. E. coli ist rot, R. eutropha blau und Z. mobilis grün.

Bei dem Vergleich mit den zusammengefassten Genomen konnten 229’802 Header Enzymen zugeordnet werden. Dies entspricht lediglich ca. 14% der gesamten Reads. 17’575 Header konnten 2 oder mehr Organismen zugeordnet werden und 3’365 Header 3 Organismen. Nach dem Grundsatz der taxonomischen Vererbung wichtiger Gene

sollten einige Übereinstimmungen zu finden sein. Da das der Fall ist bekräftigt dies die Vererbungslehre und die Relevanz der Daten. Auffallend ist, dass der gleiche Read eine Übereinstimmung mit unterschiedlichen Gensequenzen je nach Organismus hat. Beispielsweise wird Read „1854736-1“ (101 nt Lang) mit dem Protein patd (Keine EC-Nummer) und aldb (1.2.1.22) von *E. coli*, sowie Protein exac (1.2.1.3) und betb (1.2.1.8) von *R. eutropha H16* in Verbindung gebracht. Diese 3 Enzyme sind zwar alles Aldehyd Dehydrogenasen, jedoch hat *Z. mobilis pomaceae* ebenfalls Aldehyd Dehydrogenasen und hat taucht nicht in Verbindung mit diesem Header auf. Zum Einen zeigt dies die Ähnlichkeit einzelner Gensequezen zueinander auf, andererseits lässt diese Erkenntnis das Ergebnis kritischer Betrachten. Wie verlässlich ist die Aussage, wenn ein Read zu mehreren verschiedenen Enzymen passt, die sich zwar ähnlich sind, sich jedoch nicht alle ähnlichen Enzyme dieser Gruppe finden lassen. Dieses Beispiel zeigt jedoch auch die Verwandtschaft von verschiedenen Organismen, wenn sich sehr ähnelnde Sequenzen in unterschiedlichen Organismen vorhanden sind, die sogar eine ähnliche Funktion kodieren (in diesem Fall das Hydrolysieren von Aldehyden).
Eine Betrachtung des Bildes (Abbildung 5.4 zeigt die breite Abdeckung von *E. coli* und die eher Geringere von *Z. mobilis*, sowie die Unterschiedliche Expressionsstärke von einem Enzym je nach Organismus. Eigentlich wurde Angenommen, dass die Anzahl der Treffer von jedem Organismus bei einem Enzym gleich ist, da immer der gleiche Datensatz verwendet worden ist. Jedoch zeigt sich hier erneut, dass die Sequenzen nur Ähnlich, nicht jedoch Identisch sind. Das erklärt dann die unterschiedliche Stärke der Expression.
Der Multi-BLAST Lauf der 12 Genome brachte natürlich mehr Treffer zum Vorschein. Es konnten 347’144 (ca. 22%) Header Gensequenzen der einzelnen Genome zugeordnet werden. Erstaunlich ist, dass immerhin 12 Header 8 verschiedenen Genomen zugeordnet werden konnten. Ansonsten sind die gleichen Beobachtungen wie mit den zusammengefassten Genomen zu machen.
Die Maximale Anzahl an Organismen, die in dem Bild angezeigt werden, ist Standardmäßig auf 10 gesetzt. Bei dem Überfliegen von dem Bild wird empfohlen nicht mehr als sechs oder sieben verschiedene Organismen miteinander zu vergleichen, da es schnell unübersichtlich wird (siehe Abbildung 5.6).

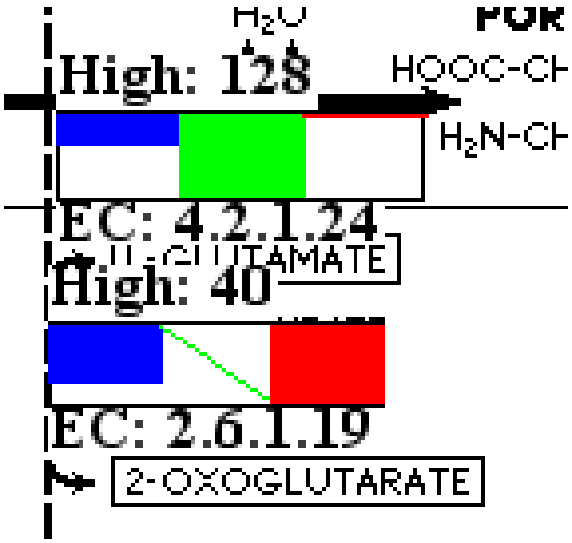

Abbildung 5.5: Ausschnitt der Stoffwechselkarte bei dem Vergleich der Zusammengefassten Genome. E. coli ist rot, R. eutropha blau und Z. mobilis grün. Ein diagonaler Strich bedeutet, dass das Enzym in dem Organismus nicht vorhanden ist. Die Höhe der Balken von Oben nach Unten gibt die Expressionsstärke im Vergleich zu der Höchsten Anzahl an Treffern bei diesem Enzym wieder.

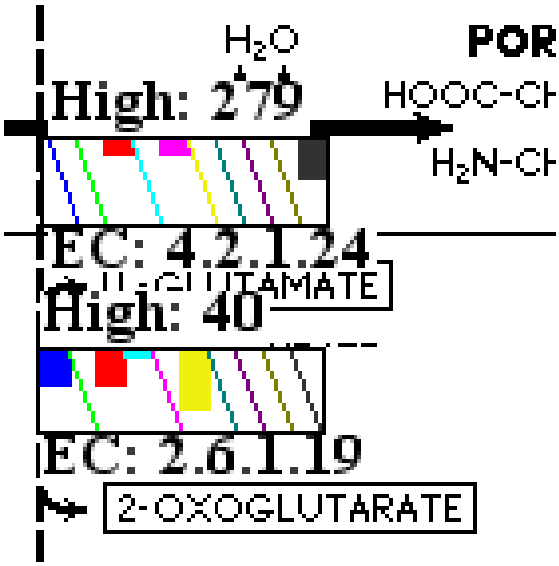

Abbildung 5.6: Ausschnitt der Stoffwechselkarte bei dem Vergleich aller untersuchten Genome. Jede Farbe entspricht einem Genom. Ein diagonaler Strich bedeutet, dass das Enzym in dem Organismus nicht vorhanden ist. Die Höhe der Balken von Oben nach Unten gibt die Expressionsstärke im Vergleich zu der Höchsten Anzahl an Treffern bei diesem Enzym wieder.

6 Ausblick

Wie jedes Programm kann auch dieses weiter optimiert werden. Einige Algorithmen können dahingehend Verbessert werden, dass sie weniger Speicher benötigen und weniger Zeit in Anspruch nehmen. Des Weiteren kann das Programm benutzerfreundlicher gestaltet werden. Wenn das Programm stabil genug läuft, kann es noch durch eine GUI Erweitert werden.
Es können weitere Ausgabeformen hinzugefügt werden, beispielsweise andere Tabellen. Außerdem kann eine genauere Analyse der Ergebnisse integriert werden, die z.B. den Anteil der Reads ausgibt, die einem Organismus zugeordnet werden können.
Es können auch weitere Datenbanken zur Suche nach Enzymen in dem Datensatz mit Eingebunden werden. Beispielsweise haben einige Genome eine SwissProt ID bei ihrem Gen vermerkt, jedoch keine EC-Nummer. Durch das Verknüpfen von 2 Datenbanken kann der SwissProt ID eine EC-Nummer zugeordnet werden.
Falls anderen Personen das Programm zur Verfügung gestellt werden soll, kann es in eine Webapplikation integriert werden, beispielsweise in das CyanoFactory Projekt.
Für die weitere Forschung sollten andere Organismen überprüft und miteinander verglichen werden. Dies hängt jedoch vor allem von der Verfügbarkeit der Genome ab.

7 Zusammenfassung

Alles im Allem war dieses Projekt fordernd und interessant. Man konnte eine stetige Verbesserung des Programmes erkennen. Anfangs wurde das Genom lediglich durch einen einfachen Identitätsvergleich mit dem Datensatz verglichen. Die Ergebnisse fielen jedoch sehr schlecht aus, so dass schnell BLAST integriert worden ist. Nun wurden weitere Funktionen, wie das Zeichen des Bildes in PNG Format, das Sortieren der Ergebnisse sowohl nach EC-Nummer als auch Anzahl der Treffer, das Zusammenfassen der Resultate nach dem Top Level Code oder nach der enzymatischen Ontologie und einige weitere kleine Features. Die hohe Komplexität erreichte das Programm jedoch mit dem erstellen des Multi-Blast Laufes, der mehrere Genome nacheinander mit dem Datensatz vergleicht und die Ergebnisse danach analysiert. Die Suche nach ähnlichen Projekten brachte die Idee mehrere offene Datenbanken miteinander zu Verknüpfen und eine PROSITE Pattern Suche zu erstellen. Diese soll den Datensatz nach Enzymen durchsuchen, die nicht in den Genomen der Organismen zu finden sind.
Die Ergebnisse verbesserten sich stetig, jedoch ist die Interpretation dieser schwierig, da noch keine vergleichbare Arbeiten veröffentlicht worden sind und die Gültigkeit der Resultate dieses Programmes dadurch nicht überprüft werden kann.

Der Autor wünscht sich, dass das Programm in einem größerem Projekt Verwendung findet und dadurch einen Beitrag für die Wissenschaft geleistet hat.

Literaturverzeichnis

[Blattner et al, 1997] Blattner, F. R.; Plunkett, G.; Bloch, C. A.; Perna, N. T.; Burland, V.; Riley, M.; Collado-Vides, J.; Glasner, J. D.; Rode, C. K.; Mayhew, G. F.; Gregor, J.; Davis, N. W.; Kirkpatrick, H. A.; Goeden, M. A.; Rose, D. J.; Mau, B.; Shao, Y. (1997) *The complete genome sequence of Escherichia coli K-12*. Science. 277:1453–1462

[Boyle and Mitchell, 1978] Boyle P.J., Mitchell R. (1978) *Absence of microorganisms in crustacean digestive tracts*. Science 200:1157–1159

[Brad Chapman und Jeffrey Chang, 2000] Brad Chapman; Jeffrey Chang (2000) *Biopython: Python tools for computational biology*. ACM Sigbio Newsletter. 20:15–19

[EnsEMBL, 2013] (24.08.2013) Ensembl release 72 - June 2013: *Ensembl genome browser 72: Homo sapiens* -. URL: `http://www.ensembl.org/Homo_sapiens/Info/Annotation#assembly`

[Fuchs, 2007] Fuchs, Georg (2007): *Allgemeine Mikrobiologie*. 8. Auflage: Georg Thieme Verlag

[Jackson, 2012] Jackson Cody (2012) *Learning to Program Using Python* 2. Auflage: Amazon Digital Services, Inc.

[King et al, 2010] King, A. J.; Cragg, S. M.; Li, Y.; Dymond, J.; Guille, M. J.; Bowles, D. J.; Bruce, N. C.; Graham, I. A.; McQueen-Mason, S. J. (2010) *Molecular insight into lignocellulose digestion by a marine isopod in the absence of gut microbes*. Proc. Natl. Acad. Sci. U.S.A. 107:5345–5350

[Madden, 2002] (29.08.3013) Madden T.: *The BLAST Sequence Analysis Tool*. (2003) IN: McEntyre J.; Ostell J.; editors. *The NCBI Handbook* (2002-) URL: `http://www.ncbi.nlm.nih.gov/books/NBK21097/`

[Mizrachi, 2007] (04.09.2013) Mizrachi, Ilene: *GenBank: The Nucleotide Sequence Database*. (2007) IN: McEntyre J.; Ostell J.; editors. *The NCBI Handbook* (2002-) URL: `https://www.ncbi.nlm.nih.gov/books/NBK21105/`

[Mougiakakos et al, 2012] Mougiakakos, D.; Okita, R.; Ando, T.; Durr, C.; Gadiot, J.; Ichikawa, J.; Zeiser, R.; Blank, C.; Johansson, C. C.; Kiessling, R. (2012) *High expression of GCLC is associated with malignant melanoma of low oxidative phenotype and predicts a better prognosis*. J. Mol. Med. 90:935–944

[Mount, 2004] Mount, David W. (2004) *Bioinformatics: sequence and genome analysis*. 2. Auflage: Cold Spring Harbor Laboratory Press

[Mozilla Developer Network et al., 2013] (02.09.2013) Mozilla Developer Network and individual contributors: *SVG | MDN*. URL: `https://developer.mozilla.org/en-US/docs/Web/SVG?redirectlocale=en-US&redirectslug=SVG`

[Roelofs, 2013] (03.09.2013) Roelofs Greg: *PNG (Portable Network Graphics) Home Site* URL:`http://www.libpng.org/pub/png/#history`

[Sigrist et al, 2013] Sigrist, C. J.; de Castro, E.; Cerutti, L.; Cuche, B. A.; Hulo, N.; Bridge, A.; Bougueleret, L.; Xenarios, I. (2013) *New and continuing developments at PROSITE*. Nucleic Acids Res. 41:D344–347

[Sigrist et al, 2002] Sigrist, C. J.; Cerutti, L.; Hulo, N.; Gattiker, A.; Falquet, L.; Pagni, M.; Bairoch, A.; Bucher, P. (2002) *PROSITE: a documented database using patterns and profiles as motif descriptors*. Brief. Bioinformatics. 3:265–274

[SQLite, 2013] (04.09.2013) SQLite: *About SQLite*. URL: `https://sqlite.org/about.html`

[Wishart, 2006] (28.08.2013)Wishart David S.: *Metabolism and Metabolic Disease Resources on the Web* IN: Valle D, Beaudet AL, Vogelstein B, Kinzler KW, et al, eds.: *Scriver's Online Metabolic and Molecular Bases of Inherited Disease*. (2011) URL: `http://dx.doi.org/10.1036/ommbid.8`

[W3C, 2010] (02.09.2013) W3C: *Secret Origin of SVG*. URL: `http://www.w3.org/Graphics/SVG/WG/wiki/Secret_Origin_of_SVG`

Bildquellenverzeichnis

[IMG-1] `https://en.wikipedia.org/wiki/File:Limnoria_4_punctata.jpg`

[IMG-2] `http://pymol.org/sites/default/files/pymol_demo_rep_small.jpg`

[IMG-3] `http://www.calvin.edu/academic/chemistry/faculty/arnoys/proteins/hemolysin1.png`

[IMG-4] `http://petang.cgu.edu.tw/Bioinfomatics/MANUALS/NCBIblast/BLAST_algorithm.gif`

[IMG-5] `http://tparslow.weebly.com/uploads/8/6/3/9/8639831/8369393_orig.jpg`

Anhang A: Programm und Ergebnisse

Das Programm und die Ergebnisse sind zu groß, als dass sie dem Anhang hinzugefügt werden können. Deshalb werden sie auf Nachfrage an die Interessierten verschickt. Bitte eine E-Mail an `mwolf3@hs-mittweida.de`.

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 14. 09 2013